# UNIVERSIDAD COMPLUTENSE DE MADRID

FACULTAD DE INFOMÁTICA

Departamento de Ingeniería del Software e Inteligencia Artificial


## TESIS DOCTORAL

# Protocolo de autoconfiguración y encaminamiento seguro para redes móviles ad hoc

MEMORIA PARA OPTAR AL GRADO DE DOCTOR

PRESENTADA POR

Julián García Matesanz

Director

Luis Javier García Villalba

**Madrid, 2013**

# Protocolo de Autoconfiguración y Encaminamiento Seguro para Redes Móviles Ad Hoc


**TESIS DOCTORAL**

*Memoria presentada para obtener el título de*
*Doctor por la Universidad Complutense de Madrid en el*
*Programa de Doctorado en Sistemas Informáticos y Programación*

**Julián García Matesanz**

*Dirigida por el profesor*
**Luis Javier García Villalba**

Departamento de Ingeniería del Software e Inteligencia Artificial
Facultad de Informática
Universidad Complutense de Madrid

Madrid, Julio de 2012

Tesis Doctoral presentada por el doctorando Julián García Matesanz en el Departamento de Ingeniería del Software e Inteligencia Artificial de la Universidad Complutense de Madrid para la obtención del título de Doctor por la Universidad Complutense de Madrid en el Programa de Doctorado en Sistemas Informáticos y Programación.

*Terminada en Madrid el 11 de Julio de 2012.*

*Título*:

**Protocolo de Autoconfiguración y Encaminamiento Seguro para Redes Móviles Ad Hoc**

*Doctorando*:

**Julián García Matesanz** (julian@sip.ucm.es)
Sección Departamental de Sistemas Informáticos y Computación
Facultad de Ciencias Matemáticas
Universidad Complutense de Madrid
28040 Madrid, España

*Director*:

**Luis Javier García Villalba** (javiergv@fdi.ucm.es)
Departamento de Ingeniería del Software e Inteligencia Artificial

Esta tesis doctoral ha sido realizada dentro del grupo de investigación GASS (Grupo de Análisis, Seguridad y Sistemas, grupo 910623 del catálogo de grupos reconocidos por la UCM) como parte de las actividades de los proyectos de investigación *Redes Ad Hoc Seguras: Sistema de Emergencias como Caso de Uso* del Programa de Fomento de la Investigación Técnica del Ministerio de Industria, Turismo y Comercio (MITyC, Programa PROFIT, referencia FIT-360000-2005-65), *Gestión de la Seguridad y Confianza en Redes Ad Hoc Híbridas* del Programa de Fomento de la Investigación Técnica del Ministerio de Industria, Turismo y Comercio (MITyC, Programa PROFIT, referencia FIT-360000-2006-64) y *Protocolo Seguro de Autoconfiguración de Direcciones IP para Redes Móviles Ad Hoc* del Programa de Tecnología Electrónica y Comunicaciones del Ministerio de Ciencia e Innovación (MICINN, Programa TEC, Subprograma TCM, referencia TEC2007-67129/TCM).

# Agradecimientos

Debo y quiero agradecer en primer lugar y muy sinceramente a Javier por haberme aceptado para realizar esta tesis doctoral bajo su dirección. Su capacidad de trabajo, su confianza y su apoyo han sido, sin duda, no sólo el acicate que me ha mantenido en el trabajo de esta tesis durante todos estos años, sino también el promotor de mi formación como investigador. Le agradezco también las facilidades y los medios que ha puesto a mi disposición para llevar a cabo las actividades y los ensayos realizados durante el desarrollo de la misma.

Quiero agradecer también muy sinceramente a Ana Lucila la ayuda valiosísima que me ha prestado de manera desinteresada a lo largo de todo el proceso de elaboración de la tesis, tanto en las pruebas realizadas como durante toda la investigación. No cabe duda de que sin su inestimable ayuda este trabajo no habría sido posible.

De la misma forma estoy enormemente agradecido a los integrantes del grupo de investigación GASS de la Universidad Complutense de Madrid porque su apoyo en todos los ámbitos de la investigación ha supuesto para mí una ayuda incalculable para la realización de esta tesis.

Y, por supuesto, el agradecimiento más profundo y sentido para mi familia. Sin su apoyo, paciencia y comprensión no habría sido posible llevar a cabo esta dura empresa. A mi esposa Isabel, a mi hija Eva y a mi hijo Sergio por su aliento.

Esta tesis doctoral ha sido realizada dentro del grupo de investigación GASS (Grupo de Análisis, Seguridad y Sistemas, grupo 910623 del catálogo de grupos reconocidos por la UCM) como parte de las actividades de los proyectos de investigación *Redes Ad Hoc Seguras: Sistema de Emergencias como Caso de Uso* del Programa de Fomento de la Investigación Técnica del Ministerio de Industria, Turismo y Comercio (MITyC, Programa PROFIT, referencia FIT-360000-2005-65), *Gestión de la Seguridad y Confianza en Redes Ad Hoc Híbridas* del Programa de Fomento de la Investigación Técnica del Ministerio de Industria, Turismo y Comercio (MITyC, Programa PROFIT, referencia FIT-360000-2006-64) y *Protocolo Seguro de Autoconfiguración de Direcciones IP para Redes Móviles Ad Hoc* del Programa de Tecnología Electrónica y Comunicaciones del Ministerio de Ciencia e Innovación (MICINN, Programa TEC, Subprograma TCM, referencia TEC2007-67129/TCM).

# Resumen

Una red móvil ad hoc es un conjunto de nodos móviles que se comunican entre sí a través de enlaces inalámbricos. Al contrario que las redes convencionales, una red móvil ad hoc no necesita la existencia de una infraestructura previa ya que cada nodo se apoya en los demás para conseguir comunicarse con otro creando la llamada comunicación multisalto. Este tipo de redes tiene varios inconvenientes que una red convencional no presenta. La topología de este tipo de redes puede cambiar rápidamente y de una forma impredecible. Además, pueden surgir variaciones en las capacidades de los nodos y enlaces, errores frecuentes en la transmisión y falta de seguridad. Por último, se deben tener en cuenta los recursos limitados de los nodos ya que normalmente una red ad hoc estará formada por dispositivos alimentados por baterías.

Los nodos de una red necesitan de algún mecanismo para intercambiarse mensajes. El protocolo TCP/IP permite comunicar a los diferentes nodos de una red asociando a cada nodo de la misma una dirección IP distinta. En redes cableadas o en redes inalámbricas con infraestructura se dispone de un servidor o de un nodo que actúa como tal que asigna correctamente las direcciones IP. En redes móviles ad hoc no se dispone de una entidad centralizada que pueda realizar esta función. Por tanto, es necesario un protocolo que realice la configuración de la red de forma dinámica y automática, que utilizará todos los nodos de la red (o sólo una parte de ellos) como si fuesen servidores que gestionan direcciones IP.

Las redes móviles ad hoc se construyen de forma dinámica cuando un conjunto de nodos crean rutas entre sí para conseguir la conectividad entre ellos. Los nodos de la red móvil ad hoc pueden actuar como origen o destino de una comunicación, pero también como encaminadores cuando una relación entre nodos no se puede realizar directamente por motivos de alcance. Un protocolo de encaminamiento de una red móvil ad hoc necesita proveer un mecanismo que mantenga las rutas hacia los destinos frente al movimiento de los nodos que puede provocar que las rutas se destruyan, y sea necesario encontrar una ruta alternativa para mantener la comunicación entre los nodos. En redes móviles ad hoc los protocolos de encaminamiento convencionales o bien tendrán un rendimiento muy pobre, o bien serán simplemente inaplicables. Como alternativa se desarrollan protocolos específicos de encaminamiento. Con frecuencia se les denomina de nivel 2.5, ya que es habitual encontrarlos por encima de protocolos de enlace como IEEE 802.11 y por debajo del protocolo de red IP.

En este trabajo se especifica un protocolo de autoconfiguración que gestiona la entrada y salida de nodos en redes móviles ad hoc. El protocolo hace que los nodos de una red móvil ad hoc colaboren entre sí para gestionar la asignación de direcciones IP únicas y correctas de forma distribuida. Todos los nodos de la red tienen el mismo papel, no existiendo un tipo de nodo especial que centralice la gestión de la misma. Los nodos cuentan con un sistema de sincronización que se apoya en el protocolo de encaminamiento OLSR. Gracias a este mecanismo la sincronización se lleva a cabo de forma pasiva, monitorizando el protocolo de encaminamiento mencionado, por lo que se genera una sobrecarga nula en el tráfico de la red con respecto al generado por el protocolo OLSR. Al ser todos los nodos responsables de gestionar la entrada de cualquier otro nuevo nodo a la red, esta operación puede realizarse rápidamente. Un nodo que desea entrar a una red intenta contactar con cualquier nodo ya perteneciente a ella, pudiendo recibir respuesta de varios nodos. Esto hace que las posibilidades de entrar a formar parte de la red con éxito sean altas, debido a la alta disponibilidad y a la redundancia que supone la gestión distribuida. El protocolo desarrollado garantiza unicidad en las direcciones IP bajo una amplia variedad

de condiciones de red que incluyen pérdida de mensajes, peticiones concurrentes y partición de redes. Cabe destacar asimismo la escalabilidad que presenta el protocolo frente a otras propuestas existentes en la literatura y su flexibilidad que facilita la extensión del protocolo con nuevas funcionalidades.

En este trabajo también se especifica una extensión del protocolo anterior más versátil: contempla la fusión de redes, resuelve la concurrencia en la inicialización de la red e incorpora nuevos mecanismos de tolerancia a fallos.

La mayoría de los protocolos de encaminamiento para este tipo de redes están diseñados sin tener en cuenta el posible comportamiento malicioso de alguno de los nodos, lo que puede ser aprovechado para vulnerar la seguridad de la red. OLSR pertenece a este grupo de protocolos donde los atacantes pueden alterar su comportamiento, de ahí la necesidad de una extensión de este protocolo.

En este trabajo también se especifica una extensión de OLSR que proporciona seguridad a OLSR ante los ataques de Generación de Mensajes Incorrectos en sus dos modalidades: Suplantación de Identidad y Suplantación de Enlaces.

Las simulaciones realizadas fueron llevadas a cabo con la herramienta *Network Simulator* (NS-3). Los resultados de las mismas demuestran que el protocolo de autoconfiguración tiene baja latencia y sobrecarga, y que su extensión presenta latencia y sobrecarga aún menores que las de su predecesor. También demuestran que la extensión de OLSR añade una ligera sobrecarga a OLSR, que apenas afecta al rendimiento, siendo una interesante alternativa para proveer integridad en OLSR frente a los mecanismos clásicos que hacen uso de criptografía, más complejos y con una gran sobrecarga.

**Palabras clave**: redes móviles ad hoc, autoconfiguración, encaminamiento, seguridad, OLSR.

# Abstract

A mobile ad hoc network is a set of mobile nodes that communicate among themselves through wireless connections. As opposed to conventional networks, a mobile ad hoc network does not need the existence of a previous infrastructure since each node relies on the others to communicate by creating the so called multi-jump communication. This type of networks have several drawbacks not found in conventional networks. For example, their topology can change quickly and unpredictably. Besides, variations in the capacities of nodes and connections may arise, frequent errors in the transmission and a lack of security. Finally, the limited resources of nodes must be taken into account, since normally an ad hoc network will contain devices fed by batteries.

The nodes of a network need a mechanism in order to exchange messages. The TCP/IP protocol allows the different nodes of a network to communicate associating a different IP to each node in it. In wired networks or in wireless networks with infrastracture a server or a node assign correctly the IP addresses. In mobile ad hoc networks there is no centralized entity capable of implementing this function. Hence, a protocol that can configure the net in a dynamic and automatic way is needed, using either all the nodes or a subset of them like servers requesting IP addresses.

The mobile ad hoc networks are dynamically built when a set of nodes create paths in order to obtain connectivity among them. The nodes in a mobile ad hoc network may act not only as origin or destinatary of a communication, but also as routers when a relationship between nodes cannot be achieved for reasons of reach. A routing protocol in a mobile ad hoc network requires providing a mechanism that maintains the paths towards the destinataries given the movement of the nodes that may cause the destruction of the paths, and that it is necessary to find an alternative route in order to keep the communication between the nodes. In mobile ad hoc networks the conventional routing protocols will either have a very poor perfomance, or will not be applicable. As an alternative, specific routing protocols have been designed. Often they are called protocols of level 2.5, since generally they are found above linking protocols like IEEE 802.11 and below the network IP protocol.

In this work an autoconfiguration protocol that manages the entry and exit of nodes in MANET networks is specified. The protocol makes the nodes in a MANET network to collaborate between them in order to manage the assignment of unique and correct IP addresses in a distributed way. All the nodes in the network have the same role, there is no special node centralizing the management of the network. The nodes have a system of synchronization based on the routing protocol OLSR. Thanks to this mechanism, synchronization is done in a passive way, controling the routing protocol mentioned above, thus generating no overload on the network traffic with respect to the one generated by the protocol OLSR. Since all the nodes are responsible in managing the entry of any new node to the network, this operation can be quickly done. A node wanting to enter the network attempts to contact any node already belonging to it, and it can receive the answer from several nodes. This makes the possibility to become part of the network quite large, due to the high disponibility and the redundancy implied by the distributed management. The developed protocol guarantees uniqueness in the IP addresses under a wide variety of network conditions, including message loss, concurrent requests and network partitions. Let's point out also the scability presented by the protocol with respect to other proposals existing in literature and its flexibility that allows the extension of the protocol to new functionalities.

In this work, also a more versatile extension of the previous protocol is specified: it

contemplates the fusion of networks, it solves the concurrence in the network initialization and it incorporates new mechanisms for fault tolerance.

Most of the routing protocols for this type of networks are designed without taking into account the malicious behaviour of one of the nodes, which may be used to vulnerate the security of the network. OLSR belongs to the class of protocols in which attackers may alter its behaviour, hence the need for an extension of the protocol. In this work, also an extension to OLSR that gives security to OLSR against attacks of Incorrect Message Generation is specified in its two forms: Identity Replacement and Node Replacement.

The simulations were performed using the tool Network Simulator (NS-3). The results show that the autoconfiguration protocol has low latency and overload, and that its extension has even lower latency and overload. They also show that the extension of OLSR adds a slight overload to OLSR, which does not affect much the performance, providing an interesting alternative for security in OLSR with respect to classical methods using cryptography, which are more complex and have a large overload.

**Keywords**: mobile ad hoc networks, autoconfiguration, routing, security, OLSR.

# Índice General

# Índice de Figuras

# Índice de Tablas

# Lista de Acrónimos

| | |
|---|---|
| AA | *Address Authority*. |
| AA | *Address Agent*. |
| ACN | *Address Conflict Note*. |
| ADVSIG | *Advanced Signature*. |
| ANSN | Advertised Neighbor Sequence Number. |
| AODV | *Ad Hoc On-Demand Distance Vector*. |
| APAC | *Agent Based Passive Autoconfiguration*. |
| AROD | *Address Autoconfiguration with Address Reservation and Optimistic Duplicated Address Detection*. |
| Autoconf WG | *Ad Hoc Network Autoconfiguration Work Group*. |
| CBR | *Constant Bit Rate*. |
| COD-OLSR | *Coded-Optimized Link State Routing*. |
| D2HCP | *Distributed Dynamic Host Configuration Protocol*. |
| DAAP | *Dynamic Address Allocation Protocol*. |
| DAD | *Duplicate Address Detection*. |
| DAD_REP | *Duplicate Address Detection Response*. |
| DAD_REQ | *Duplicate Address Detection Request*. |
| DAP | *Duplicate Address Probe*. |
| DARPA | *Defense Advanced Research Projects Agency*. |
| DHCP | *Dynamic Host Configuration Protocol*. |
| DHCP-PD | *Dynamic Host Configuration Protocol - Prefix Delegation*. |
| DNS | *Domain Name Server*. |
| DSDV | *Destination-Sequenced Distance-Vector*. |
| DSR | *Dynamic Source Routing*. |

| | |
|---|---|
| DSSS | *Direct Sequence Spread Spectrum.* |
| DYMO | *Dynamic Manet On-Demand.* |
| E-D2HCP | *Enhanced Distributed Dynamic Host Configuration Protocol.* |
| EMAP | *Extensible Manet Autoconfiguration Protocol.* |
| EPO | Ente Promotor Observador. |
| FBCB2 | *Force XXI Battle Command, Brigade-and-Below.* |
| FSR | *Fisheye State Routing Protocol.* |
| GloMo | *Global Mobile Information Systems.* |
| GPS | *Global Positioning System.* |
| GSM | *Global System for Mobile Comunications.* |
| GSR | *Global State Routing.* |
| HCQA | *Hybrid Centralized Query-Based Autoconfiguration.* |
| IANA | *Internet Assigned Numbers Authority.* |
| ICMP | *Internet Control Message Protocol.* |
| IDS | *Intrusion Detection System.* |
| IEEE | *Institute of Electrical and Electronics Engineers.* |
| IETF | *Internet Engineering Task Force.* |
| IMG | *Incorrect Message Generation.* |
| INET | *Integrated Network Enhanced Telemetry.* |
| IP | *Internet Protocol.* |
| ISM | *Industrial Scientific and Medical.* |
| ITG | *Incorrect Traffic Generation.* |
| LAR | *Location Arder Routing.* |
| LIDS | *Local IDS.* |
| LMR | *Lightweight Mobile Routing.* |
| LPR | *Low-Cost Packet Radio.* |

| | |
|---|---|
| MANET | *Mobile Ad Hoc Network.* |
| Manet WG | *Mobile Ad Hoc Networks Work Group.* |
| MID | *Multiple Interface Declaration.* |
| MPR | *Multipoint Relay.* |
| | |
| NDP | *Neighbor Discovery Protocol.* |
| NED | *Network Description.* |
| NS-2 | *Network Simulator 2.* |
| NS-3 | *Network Simulator 3.* |
| NTDR | *Near-Term Digital Radio.* |
| | |
| OFDM | *Orthogonal Frecuency-Division Multiplexing.* |
| OLSR | *Optimized Link-State Routing.* |
| OMNeT++ | *Objective Modular Network Testbed in C++.* |
| OSI | *Open System Interconnection.* |
| OSPF | *Open Shortest Path First.* |
| OSPF WG | *Open Shortest Path First IGP Work Group.* |
| oTcl | *Object-Oriented Tool Command Language.* |
| | |
| PACMAN | *Passive Autoconfiguration for Mobile Ad Hoc Networks.* |
| PAN | *Personal Area Networks.* |
| PDA | *Personal Digital Assistant.* |
| PDAD | *Passive Duplicate Address Detection.* |
| PKI | *Public Key Infrastructure.* |
| PMP | *Proactive Manet Protocol.* |
| | |
| QoS | *Quality of Service.* |
| | |
| RFC | *Request For Comments.* |
| RMP | *Reactive Manet Protocol.* |
| ROAM | *Routing On-Demand Acyclic Multi-Path.* |
| RWP | *Random WayPoint Mobility Model.* |

| | |
|---|---|
| SDAD | *Strong Duplicate Address Detection.* |
| SLAAC | *Stateless Address Autoconfiguration.* |
| SURAN | *Survivable Radio Networks.* |
| TBRPF | *Topology Broadcast Reverse Path Forwarding.* |
| TC | *Topology Control.* |
| TORA | *Temporally-Ordered Routing Algorithm.* |
| UDP | *User Datagram Protocol.* |
| VoIP | *Voice over Internet Protocol.* |
| WDAD | *Weak Duplicate Address Detection.* |
| WLAN | *Wireless Local Area Network.* |
| WRP | *Wireless Routing Protocol.* |
| WSN | *Wireless Sensor Networks.* |
| ZRP | *Zone Routing Protocol.* |

# Parte I

# Resumen de la Investigación

# Capítulo 1

# Introducción

El término *ad hoc* es una locución latina que significa literalmente ≪para esto≫. Generalmente se refiere a una solución elaborada específicamente para un problema o fin preciso y, por tanto, no es generalizable ni utilizable para otros propósitos. Se usa pues para referirse a algo que es adecuado sólo para un determinado fin. En sentido amplio, *ad hoc* puede traducirse como ≪específico≫.

El Instituto de Ingenieros Eléctricos y Electrónicos, del inglés *Institute of Electrical and Electronics Engineers* (IEEE) [MC03] define las redes ad hoc como aquellas redes compuestas únicamente por estaciones, estando cada una de ellas dentro del rango de cobertura de alguna de las otras a través de un medio inalámbrico. Una red ad hoc se crea típicamente de manera dinámica y su principal singularidad es su limitación tanto temporal como espacial. Estas restricciones permiten crear y disolver redes de manera suficientemente sencilla y práctica. Formalmente, una red ad hoc inalámbrica presenta las siguientes características [Liu05]:

- **Inalámbrica**: Los nodos o estaciones se comunican a través de medios de transmisión no guiados (radio, infrarrojos, etc.).
- **Ad hoc**: La red es temporal y se establece dinámicamente de manera arbitraria por un conjunto de nodos según se necesita.
- **Autónoma y sin infraestructura**: La red no depende de ninguna infraestructura establecida ni de ninguna administración centralizada.
- **Multisalto**: No se necesitan *routers* (encaminadores) dedicados. Cada nodo actúa como encaminador y reenvía paquetes hacia otros nodos para facilitar el intercambio de información entre los integrantes de la red.

Adicionalmente, los nodos pueden estar dotados de movilidad. En este caso, la red recibe el nombre de red móvil ad hoc, del inglés *Mobile Ad Hoc Network* (MANET). La topología de este tipo de redes es dinámica debido al constante movimiento de los nodos participantes, haciendo que los patrones de comunicación entre los miembros de la red evolucionen continuamente.

En definitiva, las redes móviles ad hoc eliminan las restricciones impuestas por las infraestructuras fijas, permitiendo a los dispositivos crear y adherirse a redes improvisadamente, haciéndolas adecuadas para adaptarse virtualmente a cualquier aplicación.

El resto de este capítulo está organizado como sigue: El apartado 1.1 presenta el objetivo de este trabajo. En el apartado 1.2 se analiza la problemática de la autoconfiguración

en redes móviles ad hoc. En el apartado 1.3 se describe el contexto en el que se ha desarrollado la investigación. Finalmente, el apartado 1.4 resume la estructura del resto de la memoria.

## 1.1 Objetivos

En una red móvil ad hoc autónoma los nodos pueden identificarse unívocamente a través de una dirección IP con la única premisa de que esta dirección sea distinta a la de cualquier otro nodo de la red.

Para comunicarse entre sí los nodos ad hoc necesitan configurar sus interfaces con direcciones locales que son válidas dentro de la red ad hoc. Los nodos ad hoc también pueden necesitar configurar globalmente direcciones de encaminamiento para comunicarse con otros dispositivos en Internet. Desde la perspectiva de la capa *Internet Protocol* (IP), una red ad hoc se presenta como una red multisalto de nivel 3 constituida por una colección de enlaces.

El proceso de configuración es el conjunto de pasos a través de los cuales un nodo consigue obtener su dirección IP dentro de la red. Existen dos mecanismos de configuración de direcciones: sin estado (*stateless*) y de estado completo (*stateful*).

La configuración de direcciones sin estado propone que sea el propio nodo el encargado de generar su dirección IP. La dirección se obtiene de la concatenación de un prefijo de red conocido y un número teóricamente único dentro de la red generada por el nodo. Este mecanismo puede exigir la inclusión de un módulo encargado de comprobar la unicidad de la dirección generada llamado Detección de Direcciones Duplicadas, del inglés *Duplicate Address Detection* (DAD).

Por otro lado, la configuración de direcciones de estado completo se basa en la utilización de servidores que controlan y asignan las direcciones a todos los nodos de la red. *Dynamic Host Configuration Protocol* (DHCP) [Dro97] es un ejemplo de configuración de estado completo. Sin embargo, dada la naturaleza multisalto de las redes móviles ad hoc, este protocolo no puede ser aplicado directamente.

Este trabajo propone un protocolo de autoconfiguración de estado completo, que garantiza unicidad en las direcciones IP bajo una amplia variedad de condiciones de red que incluyen pérdida de mensajes, peticiones concurrentes y partición de redes. Su escalabilidad y su diseño flexible permiten el desarrollo de extensiones que incorporen nuevas funcionalidades según se vayan precisando. Muestra de ello es la extensión realizada que contempla la fusión de redes, resuelve la existencia de solicitudes concurrentes de direcciones IP e incorpora nuevos mecanismos de tolerancia a fallos. Asimismo, se aborda ampliamente el problema de la seguridad en este tipo de redes, algo bastante deficiente en la literatura, proponiéndose además la extensión segura de OLSR, un protocolo de encaminamiento para redes móviles ad hoc que constituye un referente en el área y que se encuentra estrechamente relacionado con el protocolo de autoconfiguración propuesto.

## 1.2 Identificación del Problema

Las redes móviles ad hoc presentan características especiales que deben tenerse en cuenta a la hora de implementar un protocolo de configuración de direcciones. Existen muchas soluciones para redes convencionales (por ejemplo, *Request For Comments* (RFC) 3315 [DBV+03], RFC 4861 [NNSS07], RFC 4862 [TNJ07], ...) pero en su diseño no se tuvieron en cuenta las redes móviles ad hoc. Es necesario, pues, dar soporte multisalto,

soporte a topologías dinámicas y soporte a la unión (*merging*) y partición (*partitioning*) de redes, eventos que son típicos en las redes móviles ad hoc.

Hay numerosos trabajos que realizan propuestas para la configuración de direcciones en una red móvil ad hoc utilizando tanto el mecanismo sin estado como el de estado completo. Quizás la *Internet Engineering Task Force* (IETF) [IET] tenga el grupo de trabajo más conocido llamado *Ad Hoc Network Autoconfiguration Work Group* (Autoconf WG) [AUT] cuya principal finalidad es describir el modelo de direccionamiento para redes ad hoc y cómo los nodos en estas redes configuran sus direcciones. Se exige que tales modelos no causen problemas a los demás componentes de un sistema ad hoc tales como aplicaciones estándar que se ejecuten en un nodo ad hoc o nodos de Internet conectados a los nodos ad hoc. La labor de este grupo puede incluir el desarrollo de nuevos protocolos si los mecanismos de autoconfiguración IP existentes resultan inadecuados. Sin embargo, la primera tarea de este grupo de trabajo es describir un modelo de direccionamiento práctico para redes ad hoc.

Las soluciones descritas en la literatura, y que se analizan en detalle en el apartado 3.5, suponen contribuciones significativas para la comprensión del problema. Sin embargo, todas ellas manejan únicamente un subconjunto de las condiciones de red enumeradas a continuación:

1. **Topología dinámica**: Los nodos en la red se mueven arbitrariamente y pueden entrar y salir de la red dinámicamente.
2. **Pérdidas de mensajes y fallos en los nodos**: Las pérdidas de mensajes pueden ser bastante frecuentes y pueden duplicar la asignación de dirección IP si no se maneja correctamente. Los nodos pueden abandonar la red de forma abrupta debido a un fallo en el enlace o un accidente.
3. **Partición y unión de redes**: La red puede dividirse en múltiples redes y, posteriormente, unirse con otras. Durante la unión de redes es posible tener direcciones IP duplicadas en la red fusionada.
4. **Peticiones concurrentes de direcciones**: Múltiples nodos pueden querer unirse a la red simultáneamente.
5. **Energía y ancho de banda limitados**: Los nodos en una red móvil ad hoc son de energía limitada y los enlaces tienen un limitado ancho de banda. Por tanto, la sobrecarga de comunicación en la que se incurra debería ser baja.
6. **Falta de seguridad**: el diseño de protocolos para redes móviles ad hoc no suele contemplar, en la mayoría de los casos, entornos hostiles, asumiéndose la confianza de todos los nodos en la red, lo que la hace vulnerable ante diferentes clases de ataques. Consecuentemente, es común añadir extensiones de seguridad a posteriori.

En este trabajo se propone una solución que garantiza unicidad en la asignación de direcciones IP bajo un amplio conjunto de condiciones de red. En la aproximación realiada en este trabajo la mayoría de las asignaciones de direcciones implican comunicación local, originando baja sobrecarga de comunicación y baja latencia.

## 1.3 Contexto de la Investigación

Esta tesis doctoral ha sido realizada dentro del grupo de investigación GASS (Grupo de Análisis, Seguridad y Sistemas, grupo 910623 del catálogo de grupos reconocidos por la

Universidad Complutense de Madrid) como parte de las actividades de diversos proyectos de investigación que totalizan más de 5 años de trabajo.

Esta investigación se inicia en el contexto de dos proyectos de investigación del Programa de Fomento de la Investigación Técnica (PROFIT) del Ministerio de Industria, Turismo y Comercio (MITyC) de la mano de la empresa *Safelayer Secure Communications S. A.*, empresa española que constituye un referente en el área de la seguridad tanto en el ámbito nacional como internacional.

Más concretamente, esta tesis comienza con el trabajo desarrollado en los Proyectos de I+D+i *Redes Ad-Hoc Seguras: Sistema de Emergencias como Caso de Uso* (referencia FIT-360000-2005-65) [GVGMSO06] y *Gestión de la Seguridad y Confianza en Redes Ad Hoc Híbridas* (referencia FIT-360000-2006-64) [GVGMSO07].

El Programa de Fomento de la Investigación Técnica es un instrumento mediante el cual el Gobierno articula un conjunto de convocatorias de ayudas públicas, destinadas a estimular a las empresas y a otras entidades a llevar a cabo actividades de investigación y desarrollo tecnológico; según los objetivos establecidos en el Plan Nacional de Investigación Científica, Desarrollo e Innovación Tecnológica (I+D+i), en la parte dedicada al Fomento de la Investigación Técnica.

La presente investigación finaliza con otro Proyecto de I+D, en este caso con uno perteneciente al Subprograma de Comunicaciones (TCM) del Programa de Tecnología Electrónica y Comunicaciones (TEC) del Plan Nacional de I+D+i del Ministerio de Ciencia e Innovación (MICINN) titulado *Protocolo Seguro de Autoconfiguración de Direcciones IP para Redes Móviles Ad Hoc* (referencia TEC2007-67129/TCM) [GVGMSO11], en el que también participa la empresa *Safelayer Secure Communications S. A.* en este caso como Ente Promotor Observador (EPO).

## 1.4 Estructura del Trabajo

La memoria se organiza en 10 capítulos, siendo el primero la presente introducción. El resto del trabajo se estructura como sigue.

El Capítulo 2 realiza un estado del arte de las redes ad hoc incluyendo un repaso cronológico de la evolución de las redes ad hoc, un análisis de las características básicas de este tipo de redes, una presentación del protocolo de comunicación que actualmente se utiliza en ellas, una clasificación de las redes ad hoc y un resumen de las principales aplicaciones de las redes móviles ad hoc.

El Capítulo 3 aborda el problema de la autoconfiguración en redes móviles ad hoc. Comienza señalando la no aplicabilidad de las soluciones estándar. Posteriormente, presenta una clasificación de los protocolos de autoconfiguración para redes ad hoc analizando los más representativos.

El Capítulo 4 se centra en el encaminamiento en las redes móviles ad hoc. Se muestra especial atención al protocolo *Optimized Link-State Routing* (OLSR) [CP03] ya que participa activamente en el protocolo de autoconfiguración propuesto en este trabajo.

El Capítulo 5 aborda la problemática de la seguridad en las redes móviles ad hoc, haciendo énfasis en la ausencia de soluciones globales seguras pues el diseño de protocolos para este tipo de redes no suele contemplar, en la mayoría de los casos, entornos hostiles, asumiéndose la confianza de todos los nodos en la red, lo que la hace vulnerable a gran número de ataques.

El Capítulo 6 presenta las ideas básicas, los criterios de diseño y la especificación detallada del denominado *Distributed Dynamic Host Configuration Protocol* (D2HCP) o Protocolo Distribuido para la Configuración Dinámica de Direcciones en Redes Móviles

Ad Hoc, una de las principales aportaciones de este trabajo.

El Capítulo 7 presenta las ideas básicas, los criterios de diseño y la especificación detallada del denominado *Enhanced Distributed Dynamic Host Configuration Protocol* (E-D2HCP) o Protocolo Distribuido Mejorado para la Configuración Dinámica de Direcciones en Redes Móviles Ad Hoc, otra de las principales aportaciones de este trabajo, haciendo énfasis en las diferencias con respecto a su predecesor.

El Capítulo 8 presenta una extensión segura de OLSR, protocolo de encaminamiento para redes móviles ad hoc que juega un papel decisivo tanto en D2HCP como en E-D2HCP. La citada extensión, denominada *Coded-Optimized Link State Routing* (COD-OLSR), constituye otra de las principales aportaciones de este trabajo.

El Capítulo 9 contiene los resultados de las simulaciones realizadas en el software *Network Simulator 3* (NS-3) [NS3].

Por último, el Capítulo 10 muestra las principales conclusiones extraídas de este trabajo así como algunas líneas futuras de investigación.

# Capítulo 2

# Redes Móviles Ad Hoc

El objetivo general de este capítulo es facilitar la comprensión de lo que son las redes móviles ad hoc. En primer lugar se hace un repaso cronológico de la evolución de las redes ad hoc. Posteriormente, se analizan las características básicas de este tipo de redes. Luego se presta atención al protocolo de comunicación que actualmente se utiliza en este tipo de redes, el IEEE 802.11. Seguidamente, se muestra una clasificación de las redes ad hoc. A continuación, se presentan las principales aplicaciones de las redes móviles ad hoc. El capítulo finaliza con una breve síntesis de lo expuesto en el mismo.

## 2.1 Evolución Histórica

En muy pocos años el campo de las redes ad hoc ha tenido una rápida expansión visible en la proliferación de dispositivos inalámbricos de bajo coste como ordenadores portátiles, asistentes personales digitales (PDAs), teléfonos móviles, etc.

A comienzos de los años 70 un trabajo pionero en radio de la Universidad de Hawai introduce el primer sistema que usa el medio de la radio para la transmisión de información. Conocido ampliamente como ALOHA [Abr70], fue desarrollado por Abramson y Kuo.

El trabajo realizado en Hawai llevó en 1972 al desarrollo de una arquitectura distribuida consistente en una red de difusión de radio con mínimo control central llamada PARNET bajo el patrocinio de *Defense Advanced Research Projects Agency* (DARPA). El proyecto ayudó a establecer el concepto de redes móviles ad hoc. PARNET permitía la comunicación directa entre usuarios móviles sobre grandes áreas geográficas, ancho de banda compartido y protección contra los efectos de múltiples caminos.

Los rápidos avances de la tecnología de la radio en los años 70 provocó la aparición de múltiples sistemas de comunicación móvil como teléfonos celulares e inalámbricos, sistemas de radio búsqueda, satélites móviles, etc.

Posteriormente, DARPA desarrolló el proyecto *Survivable Radio Networks* (SURAN) en 1983 que trata las tareas de escalabilidad de la red, seguridad, capacidad de proceso y gestión de energía. Se dedicaron esfuerzos para desarrollar dispositivos de bajo coste y con poco gasto de energía que pudieran soportar los avanzados protocolos de encaminamiento, escalar a miles de nodos las redes y dar soporte para ataques a la seguridad. El resultado fue la aparición de la tecnología conocida como *Low-Cost Packet Radio* (LPR) en 1987.

A mitad de los 90 se produce un nuevo avance con la llegada de las tarjetas de radio 802.11 para ordenadores personales y portátiles. En [FL01] [Jai03] se propone por primera vez la idea de una colección de *hosts* móviles con una infraestructura mínima, y el IEEE acuña el término *redes ad hoc*.

Durante el mismo tiempo, el Departamento de Defensa de Estados Unidos continuaba trabajando con proyectos como el *Global Mobile Information Systems* (GloMo) o el *Near-Term Digital Radio* (NTDR). El objetivo del GloMo era permitir la conectividad multimedia de tipo Ethernet, en cualquier momento y en cualquier lugar, entre los dispositivos inalámbricos. NTDR son protocolos que se basan en dos componentes: agrupamiento y encaminamiento. Los algoritmos de agrupamiento organizan dinámicamente una red en líderes y miembros de grupo. Los líderes forman la columna vertebral de la red y los miembros se comunican entre sí a través de dicha columna. NTDR inicialmente fue un prototipo para la Armada de los Estados Unidos y en la actualidad algunos países lo utilizan como base para otros protocolos.

La definición de estándares como IEEE 802.11 [MC03] provocó el rápido crecimiento de las redes móviles en campos no sólo militares, sino también en el mundo comercial.

## 2.2 Características

Como su propio nombre indica la característica principal de una red móvil ad hoc es la movilidad de los nodos, que pueden cambiar de posición rápidamente. La necesidad de crear redes de forma rápida en lugares sin infraestructura suele implicar que los nodos exploren el área y, en algunos casos, se deban unir para conseguir un objetivo. El tipo de movilidad que desarrollen los nodos puede tener una influencia a la hora de elegir el protocolo de encaminamiento que aumente el rendimiento de la red.

Otro de los aspectos importantes en las redes ad hoc es la llamada auto-organización que se estudia en profundidad en [Fee01]. La idea principal se basa en la coordinación y colaboración de todos los nodos de la red para conseguir un mismo objetivo. Se han propuesto varios métodos de auto-organización para redes en general y para redes ad hoc en particular.

La auto-organización puede desglosarse en las capacidades mostradas en la Tabla 2.1.

Tabla 2.1: Capacidades de la auto-organización

| **Capacidad** | **Descripción** |
|---|---|
| Auto-reparación | Mecanismos que permitan detectar, localizar y reparar automáticamente los fallos siendo capaces de distinguir la causa del error. Por ejemplo, sobrecarga o mal funcionamiento. |
| Auto-configuración | Métodos de generación de configuraciones adecuadas en función de la situación actual dependiendo de las circunstancias ambientales. Por ejemplo, conectividad o parámetros de calidad de servicio. |
| Auto-gestión | Capacidad de mantener dispositivos o redes dependiendo de los parámetros actuales del sistema. |
| Adaptación | Adecuación a los cambios de las condiciones ambientales. Por ejemplo, cambio en el número de nodos vecinos. |

A continuación se presentan el resto de características de las redes móviles ad hoc:

- **Ausencia de infraestructura**: Al contrario que las redes convencionales que cuentan con la existencia de elementos físicos, las redes móviles se forman autónomamente.
- **Topología dinámica**: Los nodos se pueden mover arbitrariamente haciendo que algunos enlaces se destruyan y otros se creen cuando un nodo se acerque a otros que antes tenía fuera de su alcance.
- **Ancho de banda limitado**: En la mayoría de las ocasiones será menor que el de una conexión cableada, afectado además por las interferencias de las señales electromagnéticas.
- **Variación en la capacidad de los enlaces y los nodos**: Los nodos pueden disponer de varias interfaces de radio que difieren entre sí en capacidad de transmisión/recepción y en la banda de frecuencia en la que trabajan. Esta característica complica el desarrollo de los protocolos de encaminamiento en gran medida.
- **Conservación de energía**: Algunos o todos los nodos de una red móvil ad hoc son alimentados por baterías y no tienen posibilidad de recargarlas. Para estos nodos el criterio más importante a la hora de diseñar sistemas y protocolos será la optimización de la conservación de energía.
- **Escalabilidad**: En muchas aplicaciones las redes ad hoc pueden llegar a tener miles de nodos lo que conlleva dificultad en tareas como direccionamiento, encaminamiento, gestión de localización, gestión de configuración, interoperabilidad, seguridad, etc.
- **Falta de seguridad**: La seguridad juega un papel importante en las redes ad hoc dado el carácter vulnerable de los enlaces inalámbricos que se forman. Los protocolos de encaminamiento deben proporcionar una comunicación segura. Existen áreas de investigación en este sentido que sugieren incluir datos de sensores externos e información geográfica y topográfica en el propio algoritmo de encaminamiento.
- **Encaminamiento multisalto**: Los nodos actúan como encaminadores para retransmitir los paquetes intercambiados entre nodos cuyo alcance no permite una comunicación directa.
- **Entorno imprevisible**: Las redes ad hoc pueden darse en terrenos en los que las situaciones no son las más óptimas debido a condiciones peligrosas o desconocidas. Pueden darse casos donde los nodos se destruyan, se estropeen o comiencen a producir fallos.
- **Comportamiento de los terminales**: Una de las principales claves para que una red móvil ad hoc tenga un funcionamiento adecuado es la confianza que cada nodo debe tener sobre los demás. Sin esta confianza sería imposible crear un protocolo de encaminamiento ya que la información debe transmitirse por varios nodos intermedios. Normalmente, los protocolos de encaminamiento que descubren los terminales intermedios se basan en las respuestas que dan los nodos sobre el coste de la comunicación. Existen nodos maliciosos que podrían intencionadamente informar de forma incorrecta sobre los costes con la finalidad de recibir todos los paquetes, poder manipularlos, alterarlos o incluso eliminarlos. Algunas soluciones al respecto se encuentran en [PHM+06].

La Figura 2.1 presenta un ejemplo típico de una red móvil ad hoc.

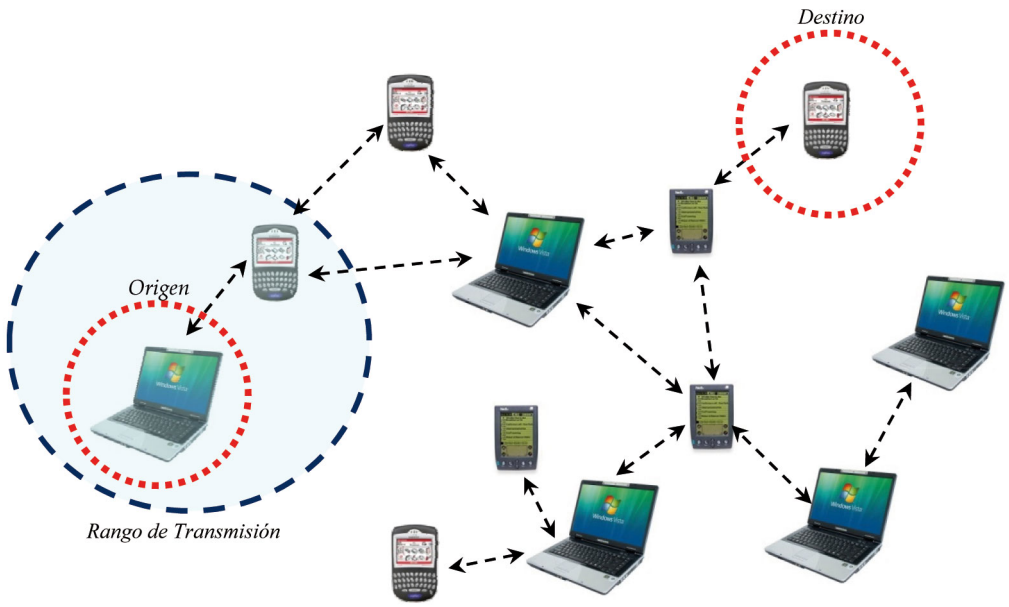



Figura 2.1: Red móvil ad hoc

## 2.3 Estándar IEEE 802.11

El IEEE 802.11 es un estándar de protocolo de comunicaciones que define el uso de los dos niveles más bajos de la arquitectura *Open System Interconnection* (OSI), capa física y capa de enlace de datos, especificando sus normas de funcionamiento en una red inalámbrica. La primera propuesta de este estándar mantenía tasas de transmisión de 1 y 2 Mbps en la banda de frecuencias *Industrial Scientific and Medical* (ISM), situada en 2.4 GHz. Además, se especificaban como tecnologías en la capa física los infrarrojos y el canal radio. Con los años se ha llegado a distintas versiones del estándar. Se citan los más importantes a continuación:

- **IEEE 802.11a**: hasta 54 Mbps a 5 GHz. Utiliza la tecnología *Orthogonal Frecuency-Division Multiplexing* (OFDM) en la capa física.
- **IEEE 802.11b**: hasta 11 Mbps a 2.4 GHz. Actualmente es el más utilizado. Utiliza la tecnología *Direct Sequence Spread Spectrum* (DSSS) en la capa física.
- **IEEE 802.11e**: pretende proporcionar *Quality of Service* (QoS) para su uso en servicios como *Voice over Internet Protocol* (VoIP) y *Streaming*. Una aproximación para otorgar calidad de servicio es la de diferenciar los paquetes clasificándolos en un número pequeño de tipos de servicios y utilizar mecanismos de prioridad para proporcionar una calidad de servicio adecuada a cada tráfico.
- **IEEE 802.11f**: desarrolla especificaciones para la implementación de puntos de acceso y sistemas de distribución para evitar problemas de interoperabilidad entre distintos fabricantes y distribuidores de equipos.

- **IEEE 802.11g**: hasta 54 Mbps a 2.4 GHz. Soporta tanto OFDM como DSSS en la capa física.

## 2.4 Clasificación

La terminología de redes ad hoc aún no está muy asentada y no existe una clasificación clara. A continuación se exponen varias clasificaciones situando el lugar en el que se encuentran las redes móviles ad hoc.

Existen redes ad hoc *con infraestructura* donde los nodos se mueven mientras se comunican con una estación base fija. Cuando un nodo se mueve fuera del rango de una estación fija entra en el alcance de otra estación. Por otro lado, se encuentran las redes ad hoc *sin infraestructura* donde no existen estaciones base fijas y todos los nodos de la red necesitan actuar como *routers*. **Las redes móviles ad hoc son redes ad hoc sin infraestructura**.

Otra clasificación de las redes ad hoc incluye las *redes de un solo salto* y las *redes multisalto*. Los nodos de las redes de un solo salto se comunican únicamente con los nodos que tienen a su alcance. En las redes ad hoc multisalto los nodos que no pueden comunicarse directamente utilizan nodos intermedios para retransmitir la información. **Las redes móviles ad hoc son redes ad hoc multisalto**.

Por último hay una clasificación que incluye las redes móviles ad hoc como un tipo independiente. Se incluyen tres tipos de redes ad hoc:

- **Redes móviles ad hoc**.
- **Redes de sensores**: También denominadas *Wireless Sensor Networks* (WSN). Formadas de dispositivos sensoriales, generalmente compuestos por un sensor tradicional y un conversor analógico-digital. La unidad de proceso está compuesta de un microprocesador y una pequeña memoria. Pueden incluir sistemas de localización y sistemas de movilidad. En estas redes el número de nodos suele ser mucho mayor que en una red móvil ad hoc pero la movilidad se considera escasa o nula (solamente cambia la topología con la pérdida o desconexión de nodos). Es habitual el flujo de información desde muchos orígenes hasta un nodo llamado sumidero (*sink*) que se encarga de procesar la información y enviarla al destino.
- **Redes híbridas**: También denominadas mixtas, son redes ad hoc que usan infraestructuras IP si están disponibles.

A su vez, las redes móviles ad hoc se pueden dividir en dos tipos en función de si están conectadas o no a otras redes:

- **Redes móviles ad hoc autónomas**: Son redes que no están conectadas a ninguna otra red. Los nodos de la red se pueden identificar unívocamente a través de una dirección IP con la única premisa de que sea distinta a la de cualquier otro nodo de la red.
- **Redes móviles ad hoc subordinadas**: Son redes conectadas a una o más redes externas. Se obliga a usar un direccionamiento IP topológico correcto y encaminable globalmente. Un ejemplo típico de red móvil ad hoc subordinada es una red móvil ad hoc que es parte de Internet.

## 2.5 Aplicaciones

Es fácil encontrar situaciones donde se ve la utilidad de las redes móviles ad hoc. Uno de los ejemplos más clásicos (aunque también discutido) es una reunión de trabajo: un grupo de personas con ordenadores portátiles o *Personal Digital Assistants* (PDAs). Son de distintas empresas y por tanto sus direcciones son distintas. Tal vez en la sala haya acceso a Internet y puedan usar por ejemplo IP móvil, pero ¿para qué pasear sus datagramas por toda la ciudad o todo el país cuando están en la misma habitación? Sus equipos probablemente estén dotados de puertos de infrarrojos o Bluetooth que les permitan formar una red para la ocasión. En algunos casos, simplemente no habrá infraestructuras de apoyo. Por ejemplo, en poblaciones aisladas o de orografía difícil, situaciones de emergencia, desastres naturales donde las infraestructuras hayan desaparecido, etc.

Otro ejemplo son las denominadas *Personal Area Networks* (PAN), redes formadas por los dispositivos de una persona, como su reloj, su agenda y su teléfono móvil. Una red así puede querer entrar en contacto con la red de otra persona que en ese momento esté próxima.

La capacidad de desplegarse inmediatamente y la no dependencia de un único punto de fallo hace a estas redes muy interesantes para el uso militar. El campo militar es posiblemente el más desarrollado actualmente. Así, el ejército estadounidense ya dispone de un sistema basado en este tipo de redes, el *Force XXI Battle Command, Brigade-and-Below* (FBCB2). Uno de sus objetivos es distinguir las fuerzas propias de las fuerzas del enemigo, ofreciendo a los soldados una visión del campo de batalla similar a la de un videojuego. Los equipos de la generación inmediatamente anterior estaban basados en comunicaciones por satélite, con latencias de cinco minutos. En abril de 2003 el FBCB2 se utilizó en la Segunda Guerra del Golfo, lo que supuso probablemente el primer uso bajo fuego real de una red móvil ad hoc.

Otro motivo por el que una red móvil ad hoc puede ser ventajosa es el coste. Aunque exista una infraestructura de red, si pertenece a una entidad ajena es muy posible que cobre por su uso, mientras que si están los equipos desplegados se dispondrá ya de una red sin coste adicional. Por ejemplo, los coches que pasan por una autopista podrían formar fácilmente una red móvil ad hoc, independientemente de su capacidad de conectarse a otras redes como *Global System for Mobile Comunications* (GSM) o similar.

Por último, supóngase que se tienen estaciones capaces de comunicarse empleando un satélite. Estos equipos de comunicaciones son caros, pero bastaría con que algunos tengan capacidad de conectarse al satélite para que todos dispusieran de conectividad. Y no todos los capaces de conectarse al satélite necesitarían estar conectados simultáneamente.

La cualidad más notable de las redes ad hoc es su flexibilidad. El hecho de que puedan establecerse en cualquier lugar y en cualquier momento sin infraestructura, administración o preconfiguración, las hace muy atractivas para un amplio rango de campos de aplicación.

La Tabla 2.2 muestra una clasificación de las aplicaciones presentes y futuras de las redes ad hoc, así como de los servicios que ofrecen [BR05].

Tabla 2.2: Aplicaciones de las redes móviles ad hoc

| | |
|---|---|
| Redes tácticas | Comunicaciones en operaciones militares. |
| | Campos de batalla automatizados. |
| Redes de sensores | Recogida de datos en tiempo real, generalmente altamente correlados en espacio y tiempo. |
| Servicios de salvamento | Operaciones de búsqueda y rescate. |
| y emergencia | Sustitución de redes con infraestructuras en situaciones de catástrofes naturales. |
| Entornos comerciales | Comercio electrónico. |
| | Oficina móvil. |
| | Servicios vehiculares. |
| Redes para particulares | *Wireless Local Area Network* (WLAN) en hogares y oficinas. |
| y redes para empresas | Redes de área personal (PAN). |
| Aplicaciones educativas | Configuración de comunicaciones ad hoc en reuniones, conferencias y congresos. |
| | Configuración de clases virtuales. |
| Ocio | Juegos multi-usuario. |
| | Robots mascota. |
| | Acceso a Internet en exteriores. |
| Servicios de localización | Servicios de seguimiento. |
| | Servicios de información. |

## 2.6 Síntesis del Capítulo

El objetivo de este capítulo ha sido introducir las redes móviles ad hoc. Se ha comenzado con un breve repaso de la evolución de las mismas. Posteriormente, se han analizado sus principales características entre las que destaca la ausencia de infraestructura, la topología dinámica y la capacidad de auto-organización. También se han comentado las distintas versiones del estándar IEEE 802.11 o protocolo de comunicación de este tipo de redes. Finalmente, se ha visto una clasificación de las redes móviles ad hoc, así como que su flexibilidad las hace idóneas para un gran número de aplicaciones.

# Capítulo 3

# Autoconfiguración en Redes Móviles Ad Hoc

Este capítulo está dedicado a la problemática de la autoconfiguración en redes móviles ad hoc, uno de los aspectos fundamentales de este tipo de redes. Se comienza viendo la necesidad del diseño de protocolos de configuración de direcciones específicos para las redes móviles ad hoc dada su naturaleza, así como las características o requisitos que deben cumplir para funcionar adecuadamente. A continuación, se comenta la imposibilidad de utilizar las soluciones tradicionales mediante dos ejemplos ilustrativos. Posteriormente, se muestra una clasificación de los protocolos de autoconfiguración según el modo de gestión de las direcciones. Seguidamente, se hace una exhaustiva revisión de estado del arte de los protocolos de autoconfiguración, presentándose ésta según la clasificación anterior. Se presta especial atención al protocolo de Mohsin & Prakash por ser un predecesor fundamental de los protocolos de autoconfiguración desarrollados en el presente trabajo. El capítulo finaliza con una breve síntesis de lo expuesto en el mismo.

## 3.1 Introducción

Los nodos de una red necesitan de algún mecanismo para intercambiarse mensajes. El protocolo TCP/IP permite comunicar a los diferentes nodos de una red asociando a cada nodo de la misma una dirección IP distinta.

En redes cableadas o en redes inalámbricas con infraestructura se dispone de un servidor o de un nodo que actúa como tal que asigna correctamente las direcciones IP.

En redes móviles ad hoc no se dispone de una entidad centralizada que pueda realizar esta función. Por tanto, es necesario un protocolo que realice la configuración de la red de forma dinámica y automática, que utilizará todos los nodos de la red (o sólo una parte de ellos) como si fuesen servidores que gestionan direcciones IP.

## 3.2 El Problema de la Autoconfiguración

Debido a la topología dinámica de las redes móviles ad hoc (constante movimiento de los nodos que pueden entrar y salir de la red frecuentemente e incluso simultáneamente), los protocolos de autoconfiguración se enfrentan a diversos problemas para garantizar la unicidad de las direcciones IP y permitir la partición y la unión de redes.

Para garantizar el correcto funcionamiento de la red los protocolos pretenden lograr los siguientes objetivos:

- **Lograr la unicidad de las direcciones IP**: Asegurar que dos o más nodos no obtengan la misma dirección IP.
- **Funcionar correctamente**: Una dirección IP está asociada a un nodo solamente por el tiempo que permanece en la red. Cuando un nodo deja la red, su dirección IP debe quedarse disponible para ser asociada a otro nodo.
- **Solucionar los problemas derivados de la pérdida de mensajes**: En caso de que algún nodo falle u ocurra pérdida de mensajes, el protocolo debe actuar lo suficientemente rápido para evitar que dos o más nodos posean la misma dirección IP.
- **Permitir el encaminamiento multisalto**: Un nodo no se configurará con una dirección IP si no hay ninguna disponible en toda la red. De esta forma, si cualquier nodo de la red posee una dirección IP libre ésta debe asociarse al nodo que está solicitando una dirección IP, aunque esté a dos o más saltos de distancia.
- **Minimizar el tráfico de paquetes adicionales en la red**: El protocolo debe minimizar el número de paquetes intercambiados entre los nodos en el proceso de autoconfiguración. En otras palabras, el tráfico de paquetes de control debe perjudicar lo menos posible al tráfico de paquete de datos ya que, en caso contrario, el rendimiento de la red disminuiría.
- **Verificar la existencia de solicitudes concurrentes de dirección IP**: Cuando dos nodos solicitan una dirección IP en el mismo instante de tiempo, el protocolo debe realizar el tratamiento pertinente para que no sea suministrada la misma dirección IP a los dos nodos.
- **Ser flexible a la partición y a la unión de redes móviles ad hoc**: El protocolo debe poder lograr la unión de dos redes móviles ad hoc distintas así como la partición en dos o más redes.
- **Realizar la sincronización**: El protocolo debe adaptarse a los rápidos cambios de la topología de las redes inalámbricas debido a la frecuente movilidad de los nodos. La sincronización se realiza periódicamente para mantener una configuración lo más actualizada posible de la topología de la red.

## 3.3 Aplicabilidad de las Soluciones Estándar

La aplicabilidad de los protocolos estándar es insuficiente para redes móviles ad hoc [Bac08]. Muestra de ello son los dos protocolos que se comentan a continuación.

*Dynamic Host Configuration Protocol - Prefix Delegation* (DHCP-PD) [TD03] es una opción de DHCPv6 [DBV+03] que provee un mecanismo para la delegación de los prefijos de direcciones IPv6 y permite la asignación automática de uno de estos. Para ello un nodo que desea obtener una dirección IPv6, envía un mensaje DHCP con la opción *prefix delegation* activada para obtener un prefijo de un servidor DHCP de la red.

La aplicabilidad en redes móviles ad hoc es insuficiente porque está basado en DHCP, por lo que asume que todos los nodos pueden conectarse ya sea directamente o a través de varios saltos con un servidor DHCP, y debido a la topología de las redes móviles ad hoc, la conexión directa al servidor DHCP no suele ser frecuente con la consecuencia de que la conexión mediante varios saltos puede producir que el servidor sea inalcanzable.

*Stateless Address Autoconfiguration* (SLAAC) [TNJ07] es un estándar que permite la autoconfiguración automática de una dirección IPv6 sin necesidad de un nodo encaminador (*router*). Para ello se ayuda del protocolo *Neighbor Discovery Protocol* (NDP) [NNSS07], un estándar para transmitir mensajes y descubrir a sus vecinos.

Un nodo crea automáticamente una dirección IPv6 uniendo su identificador de *host* (suele ser la dirección MAC) con un prefijo local conocido y realiza un proceso DAD, mediante la difusión (*broadcast*) de mensajes NDP a los vecinos.

Si la dirección IPv6 no es única el proceso de autoconfiguración se detiene y se tendrá que hacer manualmente. Si por el contrario la dirección es única, deberá pedir mediante mensajes NDP el prefijo de red y, posteriormente, volverá a comprobar con DAD si su dirección IPv6 es realmente única.

La aplicabilidad de este protocolo en redes móviles ad hoc queda limitada porque usa el protocolo NDP para enviar los mensajes y NDP asume que en la red todos los nodos están conectados entre sí. Consecuentemente, sólo soporta un único salto y lo más frecuente es que la red móvil ad hoc sea multisalto, no alcanzando a la mayoría de los nodos para realizar los procesos DAD y no pudiendo, por tanto, asegurar que la dirección IPv6 obtenida sea única.

## 3.4 Clasificación de los Protocolos de Autoconfiguración

Los protocolos de autoconfiguración pueden clasificarse dependiendo del modo de gestión de las direcciones en:

- **Protocolos de estado completo (*stateful*)**: Los nodos conocen el estado de la red, es decir, mantienen tablas con las direcciones IP de los nodos.
- **Protocolos sin estado (*stateless*)**: La dirección IP de un nodo está gestionada por el mismo. Generalmente crean una dirección aleatoria y realizan un proceso de detección de direcciones duplicadas para verificar su unicidad.
- **Protocolos híbridos**: Combinan mecanismos de los dos anteriores para mejorar la escalabilidad y la fiabilidad de la autoconfiguración. El precio es un alto nivel de complejidad en los algoritmos.

## 3.5 Trabajos Relacionados

A continuación se describen los protocolos de autoconfiguración más representativos, presentándose agrupados de acuerdo a la clasificación realizada en el apartado anterior.

Los **protocolos de estado completo** más relevantes se exponen seguidamente:

*Dynamic Address Allocation Protocol* (DAAP) [Pat01] requiere que los nuevos nodos soliciten al líder de la red la dirección IP. El líder es el nodo con la dirección IP más alta en la red. Contempla la partición y la unión de redes. El identificador único utilizado para la identificación de la red es la dirección MAC del nodo iniciador de la red. Esto podría originar un problema de múltiples redes que tengan el mismo identificador cuando el nodo iniciador se mueva fuera de la red y forme otra. No contempla la pérdida de mensajes.

MANETConf [NP02], que es una mejora de [NP01], está basado en la existencia de una tabla común distribuida a través de la cual todos los nodos son capaces de asignar direcciones IP.

Cuando un nodo quiere entrar en la red envía mensajes de difusión y al primero que le conteste lo elige como nodo iniciador y le pide una dirección IP. El nodo iniciador elige

una de las direcciones IP libres que hay en la red y antes de asignarla pide permiso al resto de nodos, porque puede darse el caso de que otro nodo la haya elegido para otro o porque puede que las tablas no estén totalmente sincronizadas debido al retardo de los mensajes.

Si la respuesta de los nodos es positiva, le asigna la dirección IP al nodo entrante y lo comunica por difusión para que el resto de nodos actualicen sus tablas.

Si hay algún nodo que no responde, se pone en contacto con él directamente (*unicast*) para obtener respuesta. Si aún así no logra obtenerla, dará por hecho que el nodo ha dejado la red, y lo comunicará al resto de nodos de la red para que actualicen sus tablas.

Para cada asignación de dirección IP esta aproximación requiere un *broadcast* a toda la red, originando problemas de escalabilidad. Sin embargo, esta aproximación gestiona la partición y unión de redes.

El esquema proactivo de Mohsin y Prakash [MP02], que es una evolución de [MM00] [MDMD01], trata de solucionar el problema de la asignación de direcciones IP mediante la división binaria de bloques de direcciones libres.

Estas divisiones se hacen en potencias de 2. De esta forma pueden existir nodos con varios o ningún bloque. Además, cada nodo de la red dispone de una tabla con el estado de todos los nodos de la red, es decir, conoce los bloques de direcciones libres de los otros nodos, así como la dirección IP de la interfaz de red de cada nodo.

El proceso de asignación de direcciones puede proceder de cualquier nodo. Siguiendo esta idea, un nodo no configurado (nodo cliente) envía un mensaje de difusión del tipo *REQUEST* para que un nodo de la red lo configure. Al ser un mensaje de difusión podrían producirse varias respuestas, pero elegirá al nodo que realice la primera respuesta (*REPLAY*), y éste actuará como nodo servidor. Si este nodo dispone de varios bloques, le entregará uno al nodo cliente, y elegirá la primera IP del bloque como propia. Si por el contrario dispone de un bloque, entonces lo dividirá en 2 partes iguales y le entregará una mitad al nodo cliente, y la otra se la quedará como propia.

Si se diera el caso de que no dispusiera de ningún bloque se proponen ciertas soluciones basadas en la idea de que el servidor busque en sus vecinos próximos si alguno tiene algún bloque disponible. En caso de no encontrar direcciones libres intentará obtener un bloque en vecinos de un salto mayor, y así sucesivamente. Otra solución es buscar el nodo que tenga mayor rango de direcciones libres, para entregar la mitad de este bloque al nodo que está entrando en la red.

La entrada y salida de los nodos puede ser de forma abrupta o voluntaria, para cada caso hay un esquema a seguir. Si la salida es abrupta, el nodo que actuó como servidor en su configuración verá en su tabla de rutas que el bloque del nodo que abandonó la red no está y añadirá el bloque a su bloque de direcciones libres. En el caso de una salida voluntaria, el nodo que sale notifica a algún vecino suyo la intención de marcharse y el vecino busca al nodo que configuró al cliente para que se quede con su bloque.

La mayor ventaja de este protocolo es que funciona bien para unión y división de redes, ya que soluciona el problema de las direcciones duplicadas que se producen en estos casos. Cada nodo que inicia la red genera un número aleatorio llamado *PartitionID* que será un número identificativo de la red. De esta manera cuando se produce una división o partición de la red, el primer nodo que se retire de la red original creará otro *PartitionID*. En el momento en que dos redes con *PartitionID* diferentes se unan, primero se comprobará la consistencia de sus direcciones IP. Este proceso consiste en comprobar la existencia de dos nodos con una misma dirección. En caso afirmativo se produce un cambio del nodo perteneciente a la red con menor rango de direcciones IP libres. La nueva dirección IP pertenecerá al rango de direcciones de la red con mayor número de direcciones libres.

El mayor inconveniente de este protocolo es que la sincronización depende de la exis-

tencia de un *broadcast* confiable y en un entorno distribuido móvil tal cosa no existe, por lo que uno puede plantearse dudas sobre la robustez de este protocolo.

[ZNM03] propone una solución (denominada *Prophet Address Allocation*) que se deriva del esquema de generación de una secuencia que consta de números dentro de un rango $R$ que se genera utilizando una función *f(n)*. Esta función se elige de tal forma que en las secuencias generadas por ella, el intervalo entre dos ocurrencias del mismo número sea muy grande y la probabilidad de que más de una ocurrencia del mismo número en un número limitado de secuencias diferentes iniciadas por distintas semillas sea extremadamente baja. El protocolo requiere que el primer nodo elija un número aleatorio del rango $R$ como su dirección IP y utilice un valor de estado aleatorio como semilla para la función *f(n)*. Cuando un nuevo nodo se une a la red, el nodo configurado genera otro entero y un nuevo valor de estado usando *f(n)*. El nuevo nodo obtiene estos valores y se configura a sí mismo. En esta aproximación el bloque requerido de direcciones IP puede ser significativamente más grande que el número de nodos en la red. Incluso si el intervalo mínimo entre dos ocurrencias de la misma dirección en la secuencia es extremadamente grande, todavía es posible que dos nodos tengan la misma dirección IP si los nodos entran y salen de la red con elevada frecuencia.

Una mejora del esquema de Mohsin y Prakash (particularmente en lo relativo a la sincronización) puede encontrarse en [TP06], donde se propone una asignación dinámica de direcciones basada en el denominado *buddy system* (sistema de amigos) que maneja movilidad de los nodos durante la asignación de direcciones, la pérdida de mensajes y la partición y unión de redes. Sin embargo, la asignación de direcciones IP puede generar alta sobrecarga de mensajes de control mientras hace una búsqueda global y la recuperación de direcciones (para evitar pérdidas de direcciones) requiere de mensajes de difusión por inundación (*flooding*). Además, la unión y partición pueden incurrir en una alta sobrecarga debido a la naturaleza global de este protocolo.

El protocolo de autoconfiguración presentado en [STC08] propone un esquema donde los nodos se clasifican en nodos coordinadores y nodos comunes. El primer coordinador que inicia la red se denomina C-raíz (coordinador raíz). Los coordinadores gestionan el rango de direcciones IP y son responsables de la asignación de una dirección IP a un nodo que acaba de unirse a la red. Los nodos que desean unirse a la red intercambiarán mensajes *Hello* para encontrar al nodo coordinador más cercano y obtener una nueva dirección IP del nodo coordinador. Para gestionar el rango de direcciones IP eficientemente, los nodos coordinadores están distribuidos en una topología de árbol denominada C-árbol (árbol de coordinadores) intercambiándose mensajes *Hello*. Este protocolo no considera ni la partición ni la unión de redes.

Los **protocolos sin estado** hacen uso del mecanismo DAD [PMW+01], proceso que utilizan los protocolos para comprobar la unicidad de las direcciones IP. Este proceso requiere un tiempo relativamente largo para completarse, por lo que se han implementado diferentes soluciones que lo reducen. Existen tres tipos de procesos DAD:

- ***Strong Duplicate Address Detection* (SDAD)**: SDAD [PMW+01] es la base de los protocolos sin estado. Consiste en un sencillo mecanismo en el que el nodo elige dos direcciones IP, una temporal y una tentativa. La dirección temporal sólo la usará para la inicialización mientras detecta si la tentativa es única. El método de detección consiste en enviar un mensaje *Internet Control Message Protocol* (ICMP) destinado directamente a esa dirección. Si recibe respuesta, esa dirección IP está siendo usada por lo que reanudará el proceso. Si no recibe respuesta, enviará el mensaje un número determinado de veces para asegurarse de que es única. Al ser un mecanismo muy sencillo, no asegura la unicidad de la dirección IP ya que el proceso

se limita sólo a la fase de inicialización, y para desconexiones temporales o pérdida de la red no funcionaría. Además, cuando la red es grande y quedan pocas direcciones IP libres, añade mucha sobrecarga hasta que encuentra una dirección IP única. Esta aproximación requiere que el protocolo de encaminamiento tenga una fase de descubrimiento de rutas. No aborda el problema de la partición de redes.

- ***Weak Duplicate Address Detection* (WDAD)**: WDAD [Vai02] establece la idea de tolerar durante un tiempo las direcciones duplicadas en la red. Para ello cada nodo al iniciarse creará una clave que enviará siempre junto a su dirección IP. Cuando un nodo reciba un mensaje comprobará en su tabla si esa dirección IP ya está asignada y mirará si las claves coinciden; si no coinciden, marcará esa dirección como inválida y se tomarán acciones para que sean únicas (estas acciones no están definidas en WDAD).

  Este proceso tiene que soportar la identificación de un nodo mediante un par clave-IP y depende totalmente del protocolo de encaminamiento, ya que se requiere modificar el protocolo de encaminamiento en lo relativo a los paquetes de control para transportar la información de la clave. Sólo funcionará con un protocolo proactivo que actualiza las rutas constantemente, pero con un protocolo reactivo habrá nodos que nunca puedan detectar la duplicidad de direcciones IP.

  No añade sobrecarga adicional al protocolo de encaminamiento, pero en cambio si añade la sobrecarga de enviar siempre la clave junto a la dirección IP.

- ***Passive Duplicate Address Detection* (PDAD)**: En PDAD [Wen03] la idea se basa en que en vez de detectar o resolver direcciones IP duplicadas enviando información de control, cada nodo investiga y deduce si existe una dirección duplicada por eventos que nunca ocurrirían si todas las direcciones IP fueran únicas.

  Se proponen tres detecciones pasivas, que son necesarias unir para el correcto funcionamiento de la detección:

  - **PDAD-SN (Sequence Numbers)**: Este sistema se basa en la idea de que los protocolos de encaminamiento usan números de secuencia en sus mensajes para actualizar las rutas. Usando estos números de secuencia, y la idea de que dos nodos con una distancia entre ellos de dos saltos no tienen el mismo vecindario, se solucionan algunos conflictos. Además, se tiene en consideración la posibilidad de que estos números de secuencia lleguen al máximo y empiecen a producirse números desde cero de nuevo.
  - **PDAD-LP (Locality Principle)**: De menor potencia que el anterior, se basa en la frecuencia de actualización de las tablas de rutas. En función de esta frecuencia se pueden detectar direcciones duplicadas tomando un umbral de tiempo para visualizar el estado de las tablas de rutas. Hay que tener en cuenta el protocolo de encaminamiento usado. Deben considerarse diferentes umbrales, ya que se puede dar el caso de que dos mensajes con el mismo origen se confundan con una dirección duplicada si el tiempo es demasiado corto (el protocolo modifica las rutas demasiado rápido).
  - **PDAD-NH (Neighborhood)**: Otra posibilidad para detectar direcciones duplicadas es explotar la propiedad de que un nodo conoce a sus vecinos y a los vecinos del origen de un paquete de estado de enlace. Si la dirección de A es única, un nodo con la dirección A sólo recibe paquetes conteniendo la dirección A si el origen fue vecino del nodo A en el instante en el que se envió el paquete.

Si este no es el caso, existe un conflicto de direcciones. Consecuentemente, los nodos deben mantener una caché con las direcciones de los vecinos recientes. Sin embargo, en el caso de que el remitente del paquete de estado de enlace sea un vecino común a los nodos con la misma dirección, el conflicto no puede ser detectado por PDAP-NH. Por tanto, conflictos en vecinos de dos saltos deben detectarse por otros medios. Un reto es elegir un valor de *timeout* óptimo para las entradas de caché. Si el *timeout* es demasiado alto, la memoria se infra-utiliza y algunos conflictos pueden permanecer sin detectarse. Si es demasiado bajo, los nodos detectan erróneamente conflictos. El valor depende del tiempo máximo que un mensaje puede circular por la red.

La ventaja es que no añade sobrecarga a la red adicional, pero solamente se puede utilizar con protocolos de encaminamiento proactivos.

La autoconfiguración sin estado de IPv6 [TN98] [TNJ07] especifica los pasos de un nodo que quiera configurar sus interfaces en IPv6. Los pasos incluyen la construcción de una dirección local de enlace, detección de direcciones duplicadas y construcción de una dirección local de sitio. La detección de direcciones duplicadas en redes móviles ad hoc requiere de difusión, lo que hace que esta aproximación no sea escalable.

Para abordar el problema de la escalabilidad en [WZ02] se propone una extensión construyendo una estructura jerárquica. Pero el coste que requiere mantener tal estructura jerárquica puede ser elevado.

El esquema de autoconfiguración de direcciones sin estado presentado en [JCPK03] consta de tres fases: (1) selección de dirección aleatoria, (2) verificación de la unicidad de la dirección y, (3) asignación de la dirección a la interfaz de red. La verificación de la unicidad se hace por un esquema DAD híbrido de dos fases: (a) fase de SDAD y, (b) fase de WDAD. Dentro de una red ad hoc conectada un nodo se configura a sí mismo con una dirección IP utilizando SDAD. Durante esta fase el nodo elige una dirección tentativa y comprueba si está duplicada enviando el mensaje *AREQ* por *broadcast* con la dirección tentativa elegida durante un número determinado de veces. Si no recibe respuesta al mensaje AREQ, el nodo se configura a sí mismo con la dirección tentativa. WDAD utiliza una clave además de la dirección IP para detectar direcciones duplicadas durante el encaminamiento ad hoc.

Este esquema asegura que durante la resolución de un conflicto de direcciones, las sesiones que utilizan direcciones conflictivas son mantenidas hasta que se cierran.

[CAG05] describe un método de autoconfiguración del *host* eligiendo aleatoriamente una dirección local de enlace dentro del rango 169.254.1.0 - 169.254.254.255. Después de seleccionar la dirección, el *host* comprueba si la dirección está en uso por otro nodo. Esta aproximación se centra en redes cableadas y asegura la unicidad de la dirección local de enlace. Se requiere que cada nodo en la red esté dentro del rango de comunicación de los otros nodos, lo cual no siempre es posible en el caso de una red móvil ad hoc. Para extender la solución a redes móviles ad hoc, los mensajes de detección de conflicto tendrán que ser difundidos por inundación (*flooding*) por la red.

AIPAC [FVP06] es un protocolo que utiliza una aproximación reactiva en la asignación de direcciones IP, por lo que tiene que gestionar direcciones duplicadas. Se centra en mantener la unicidad de direcciones después de la unión de redes consecuencia de la movilidad de los nodos. El protocolo tiene como prioridad el soporte de las siguientes características de las redes ad hoc: recursos limitados de los dispositivos y la no fiabilidad de los canales inalámbricos.

Cada red se reconoce por su identificador de red (NetID). Cuando dos redes se unen de forma persistente, sus identificadores deben unificarse. Para lograrlo utiliza un mecanismo

de fusión gradual. Este permite a un nodo pasar de un identificador de red a otro según los cambios en la red observados. Este procedimiento permite tener un sistema homogéneo en el caso de múltiples redes solapadas según la evolución de sus topologías.

Este protocolo no garantiza la unicidad de las direcciones IP asignadas, pero asegura que los mensajes son encaminados correctamente. Cada nodo en AIPAC es consciente únicamente de su vecindario, limitando a éste la información almacenada por el nodo.

[RRP06] propone *Extensible Manet Autoconfiguration Protocol* (EMAP), un protocolo de autoconfiguración que se basa en la idea del protocolo de mensajes *Request/Replay* para su funcionamiento. La principal ventaja de este protocolo es la posibilidad de hacerlo extensible, es decir, que en el futuro se pueden incluir nuevas funcionalidades que quedan tratadas de forma teórica como el descubrimiento de servidores *Domain Name Server* (DNS). Este protocolo trata también la posibilidad de comunicaciones exteriores a la red móvil ad hoc a través de Internet. El mecanismo de descubrimiento de rutas entre nodos sigue la línea del protocolo *Ad Hoc On-Demand Distance Vector* (AODV) [PBRD03].

La principal idea de este protocolo de autoconfiguración es la de poseer diferentes direcciones por parte de un nodo no configurado que va a unirse a la red ya creada. De esta manera existen tres direcciones para las comunicaciones interiores: la dirección temporal (*temporary address*), la dirección tentativa (*tentative address*) y la dirección local de red móvil ad hoc (*mobile ad hoc network local address*).

Cuando un nodo quiere entrar en la red genera aleatoriamente dos direcciones IP válidas de la red (con dirección de red conocida) y las considera como dirección temporal y dirección tentativa. Estas direcciones IP se encapsulan en el mensaje *Duplicate Address Detection Response* (DAD_REP) para saber si es una dirección válida. El nodo queda esperando un *Duplicate Address Detection Request* (DAD_REQ). Si transcurre el tiempo de espera para este mensaje el nodo asume que puede usar su dirección tentativa como única y se la asigna a su interfaz de red. Si este nodo recibe un mensaje DAD_REP en su dirección temporal y este mensaje contiene el origen con la dirección tentativa que había propuesto, sabe que esta dirección tentativa está siendo usada y comienza nuevamente el proceso anterior creando otro par de direcciones.

Esta aproximación requiere una adaptación de los protocolos de encaminamiento para redes móviles ad hoc existentes. No contempla explícitamente la unión de redes.

*Agent Based Passive Autoconfiguration* (APAC) [LCXL07] es un protocolo de autoconfiguración basado en PDAD. Su característica principal es el uso de ciertos nodos que centralizan el reparto de direcciones.

El mecanismo por el que un nodo configura su dirección IP al entrar en la red consiste en preguntar si tiene a un salto de distancia algún nodo del tipo *Address Agent* (AA). En este caso, el nodo AA le proporcionará una dirección IP.

En caso de no tener respuesta de ningún nodo AA, el nodo entrante se configura para funcionar en modo AA y hará de servidor de direcciones para los próximos nodos que lleguen. Cuando se configura como AA, el nodo genera aleatoriamente un número identificador, *agentID*, para poder formar una tabla con las direcciones que asignará a los nodos que lleguen. Esas direcciones son de la forma *agentID+hostID*.

Cuando un nodo se mueve en la red y sale del radio de cobertura del AA que le proporcionó su dirección IP, deberá pedir otra dirección a otro nodo AA que tenga dentro de su nuevo radio de cobertura. Esto se complementa con un mecanismo para no interrumpir las comunicaciones en curso. Al darse esta situación, el AA anterior marcará su dirección IP como libre para poder ser asignada a algún otro nodo más adelante.

La detección de direcciones duplicadas se realiza mediante el proceso PDAD. Una vez detectado algún conflicto, se informa al AA que asignó esa dirección IP conflictiva. Este

nodo AA entonces generará un nuevo *agentID* y avisará a todos los nodos dependientes de él para que cambien su dirección del tipo *agentID+hostID* al nuevo *agentID*.

En cuanto a la división y unión de redes se utiliza el mismo mecanismo explicado anteriormente. En caso de división, los nodos AA marcarán las direcciones que hayan salido de la red como libres. Y en el caso de unión de dos redes, el mecanismo para la detección de direcciones duplicadas sigue funcionando correctamente.

En *Address Autoconfiguration with Address Reservation and Optimistic Duplicated Address Detection* (AROD) [KAL07] la reserva de direcciones se basa en la existencia de unos nodos que tienen una dirección IP reservada para entregársela a los nodos que entren nuevos. Existirán dos tipos de nodos:

- Agentes tipo 1 con una dirección IP reservada, aparte de las direcciones IP que tengan sus interfaces de red. Cuando un nodo entra en la red, se le asignará esta IP reservada inmediatamente.
- Agentes tipo 2, que no tienen direcciones IP reservadas. Si un nodo entrante le pide una dirección IP a uno de éstos, pide prestada la dirección reservada de uno de sus vecinos que sea de tipo 1 y se la asigna al nuevo inmediatamente.

AROD establece un mecanismo para que la red no se quede sin nodos de tipo 1. Cada vez que se asigna una dirección IP a un nuevo nodo, el nodo que ha entregado la dirección genera dos direcciones IP aleatorias (una para sí mismo y otra para el nuevo nodo que ha entrado) y se hace un proceso DAD para detectar si esas direcciones son únicas. Una vez realizado el proceso se pueden dar las siguientes posibilidades:

- Las dos direcciones IP son únicas, por lo tanto, los dos nodos se convierten en nodo tipo 1.
- Si sólo una es única, se convertirá en nodo tipo 1 el que dio la dirección IP.
- Si ninguna es única, los dos nodos se quedarán como tipo 2.

Este protocolo considera su proceso de detección de direcciones duplicadas como un proceso DAD optimista, porque solamente se lleva a cabo una vez (cuando entra un nodo y se le asigna una dirección IP reservada).

AROD contempla la posibilidad de cambiar el número de direcciones IP reservadas para los nodos tipo 1. Si aumenta el número, la latencia de asignación de direcciones IP es menor, pero por el contrario aumenta la sobrecarga al tener que realizar más procesos DAD, y viceversa. Esta posibilidad permite variar latencia versus sobrecarga.

Finalmente, cabe señalar los siguientes protocolos híbridos:

*Hybrid Centralized Query-Based Autoconfiguration* (HCQA) [YSBR03] [WZ04] fue el primer protocolo de autoconfiguración híbrido.

Un nodo que quiera entrar en la red realiza un proceso SDAD. Si el proceso es exitoso, el nodo deberá registrar su dirección IP tentativa en una *Address Authority* (AA). Para ello, esperará un mensaje de la AA y cuando lo reciba enviará una petición de registro y la AA se lo confirmará. El nodo al comenzar todo este proceso inicia un contador; si el contador expira, volverá a empezar de nuevo el proceso hasta poder registrar la dirección IP.

Cuando se crea la red, el primer nodo se convierte en AA, elige un identificador para la red único (por ejemplo la dirección MAC) y mediante mensajes de difusión la anuncia periódicamente para identificar la red. Si algún nodo no la recibe se considera que la red ha sido dividida y creará su propia red convirtiéndose en AA.

Este protocolo añade robustez al proceso SDAD y garantiza la no duplicidad de las direcciones IP y a la vez proporciona un buen mecanismo de partición de redes.

Pero tiene dos problemas: primero, la sobrecarga producida por el proceso SDAD y los mensajes periódicos de la AA, y segundo, que la red depende de una entidad central con la que todos los nodos deben comunicarse directamente para poder registrar su dirección IP, con lo que se añade mucha latencia en la entrada de los nodos a la red.

*Passive Autoconfiguration for Mobile Ad Hoc Networks* (PACMAN) [Wen05] es un protocolo de autoconfiguración pasivo para redes móviles ad hoc. Utiliza elementos de protocolos con y sin estado, por lo que puede considerarse híbrido en cierto modo. Su funcionamiento se basa en que cada nodo se asigna a sí mismo una dirección al entrar a la red, y en la monitorización pasiva de las comunicaciones para la detección de direcciones duplicadas. Para conseguir la mínima sobrecarga en las comunicaciones se comparte información entre las diferentes capas de red. En concreto, se monitoriza la información que maneja el protocolo de encaminamiento. El método usado para elegir la propia dirección IP que se usará sigue un algoritmo probabilístico. Para intentar que la probabilidad de elegir una dirección IP actualmente en uso por otro nodo sea cercana a cero, este algoritmo tiene en cuenta, entre otros factores, una tabla de asignaciones. Esta tabla se crea con información extraída del protocolo de encaminamiento sobre las direcciones IP que ya están en uso.

PACMAN usa el proceso PDAD para monitorizar las comunicaciones en busca de direcciones duplicadas. Esto es necesario debido a que el mecanismo utilizado para la asignación de direcciones no garantiza unicidad (aunque intenta reducir la probabilidad de colisión), y a que pueden producirse uniones de redes que contengan nodos con las mismas direcciones IP. A grandes rasgos hay dos tipos de eventos que indican duplicidad de direcciones IP: Por un lado están los eventos que nunca ocurren si la dirección es única, y siempre si la dirección está duplicada. Estos eventos confirman que hay un problema detectado. Y por otra parte están los eventos que ocurren pocas veces si la dirección es única, y a menudo si la dirección está duplicada. De este modo se detecta la posibilidad de que existan problemas, por lo que se trata de algoritmos probabilísticos.

Cuando se detecta que dos nodos están usando la misma dirección IP, se le notifica el problema a uno de ellos mediante un mensaje *unicast* para que cambie su dirección.

Además, se tiene en cuenta el problema que surge cuando se cambia una dirección que tiene alguna comunicación en marcha. Para solucionarlo, al cambiar de dirección un nodo avisa a los nodos con los que tiene comunicaciones en curso de su nueva dirección IP para que hagan un encapsulamiento de los mensajes adecuadamente.

Bernardos *et al.* [BCM08a] [BCM08b] [BCM08c] realizan un riguroso estudio de la problemática de la autoconfiguración en redes móviles ad hoc, presentando una revisión detallada de los protocolos de autoconfiguración más representativos.

Las soluciones descritas anteriormente han supuesto contribuciones significativas para la comprensión del problema. Sin embargo, todas estas aproximaciones manejan únicamente un subconjunto de las condiciones de red enumeradas a continuación:

1. **Topología dinámica**: Los nodos en la red se mueven arbitrariamente y pueden entrar y salir de la red dinámicamente.

2. **Pérdidas de mensajes y fallos en los nodos**: Las pérdidas de mensajes pueden ser bastante frecuentes y pueden duplicar la asignación de dirección IP si no se maneja correctamente. Los nodos pueden abandonar la red de forma abrupta debido a un fallo en el enlace o un accidente.

3. **Partición y unión de redes**: La red puede dividirse en múltiples redes y, posteriormente, unirse con otras. Durante la unión de redes es posible tener direcciones IP duplicadas en la red fusionada.

4. **Peticiones concurrentes de direcciones**: Múltiples nodos pueden querer unirse a la red simultáneamente.

5. **Energía y ancho de banda limitados**: Los nodos en una red móvil ad hoc son de energía limitada y los enlaces tienen un limitado ancho de banda. Por tanto, la sobrecarga de comunicación en la que se incurra debería ser baja.

En este trabajo se propone una solución similar a [MP02] que garantiza unicidad en la asignación de direcciones IP bajo un amplio conjunto de condiciones de red. En la aproximación realizada, la mayoría de las asignaciones de direcciones implican comunicación local originando baja sobrecarga de comunicación y baja latencia.

### 3.5.1 Protocolo de Mohsin & Prakash

[MP02] se trata de un protocolo de autoconfiguración para redes móviles ad hoc que garantiza la unicidad de las direcciones IP mediante el proceso de división binaria de direcciones. En este protocolo todos los nodos de la red son encargados de asignar direcciones IP a los nuevos nodos que entran. Cada uno tiene asignadas una serie de direcciones IP que puede asignar. También es responsabilidad de todos los nodos el mantener actualizada la información sobre las direcciones IP libres disponibles propias y del resto de nodos de la red. Mediante un sistema de sincronización simple los nodos son capaces de mantener unicidad en las direcciones IP asignadas, garantizando la no duplicidad. El protocolo especifica cómo actuar en los eventos de entrada y salida de nodos de la red, así como en la partición y fusión de redes.

#### Estructuras de Datos

Cada nodo debe mantener dos tablas que se irán actualizando constantemente mediante un mecanismo de sincronización para poder reflejar el estado de la red actual.

- **Free_IP_Blocks**: Tabla que contiene los bloques de direcciones IP libres para atender a los nuevos nodos. Cada bloque almacenado debe cumplir la condición de ser disjunto del resto de bloques de la red. En esta tabla también se almacenará la dirección IP del propio nodo.

- **Allocated_IP_blocks**: Tabla que almacena la topología de la red. Cada entrada de la tabla se refiere a un nodo, almacenando los siguientes datos:

  - *IP Address*: Dirección IP asignada a la interfaz.
  - *Free_IP_blocks*: Bloques de direcciones IP libres (indicando en qué bloque se encuentra la dirección IP de la interfaz).
  - *TimeStamp (TS)*: Antigüedad del nodo, necesaria a la hora de recuperar direcciones que algún nodo deja disponibles al abandonar la red.
  - *Father*: Dirección IP del nodo que hizo de servidor cuando entró a la red.

Además cada nodo debe conocer el identificador de la red, *Partition ID* (PID), generado por el nodo que creó la red. Este PID es necesario en la partición y unión de redes.

### Mensajes

En este protocolo se utilizan 7 mensajes durante el proceso de entrada de un nodo a la red:

- **ADDR_REQ**: Mensaje que el nodo cliente envía al servidor para solicitar una dirección IP.
- **ADDR_REP**: Respuesta del servidor al nodo cliente cuando recibe el mensaje de solicitud de dirección IP.
- **SERVER_POLL**: El nodo cliente envía este mensaje a un nodo servidor para indicarle que lo eligió como servidor para la configuración de su dirección IP.
- **IP_ASSIGNED**: Mensaje enviado por el nodo servidor al nodo cliente asociando la dirección IP.
- **IP_ASSIGNMENT_OK**: Mensaje enviado por el nodo cliente al nodo servidor confirmando que el proceso de autoconfiguración fue realizado con éxito.
- **NODE_UP**: Mensaje enviado para notificar la entrada de un nuevo nodo, de forma que el resto de ellos puedan actualizar las tablas.
- **NODE_UP_REPLY**: Mensaje de confirmación de actualización de tablas tras la entrada de un nodo.

Asimismo, para detectar la salida de un nodo se utilizan los 3 mensajes siguientes:

- **GRACEFULL_DEPARTURE**: Mensaje enviado para entregar las direcciones libres que deja un nodo al salir avisando a la red.
- **NODE_DOWN**: Mensaje que se envía a todos los miembros de la red para notificarles la desaparición de un nodo.
- **NODE_DOWN_REPLY**: Mensaje de respuesta al *NODE_DOWN* para indicar que los cambios en las tablas han sido realizados.

### Funcionamiento General

Cada nodo almacena en sus estructuras de datos, entre otros, una serie de bloques de direcciones IP libres. Estos bloques son la base fundamental en la que se sostiene el mecanismo de asignación de direcciones por división binaria.

La Figura 3.1 muestra cómo se realiza la división binaria de esos bloques de direcciones libres, partiendo de un nodo que creó la red con 256 direcciones IP.

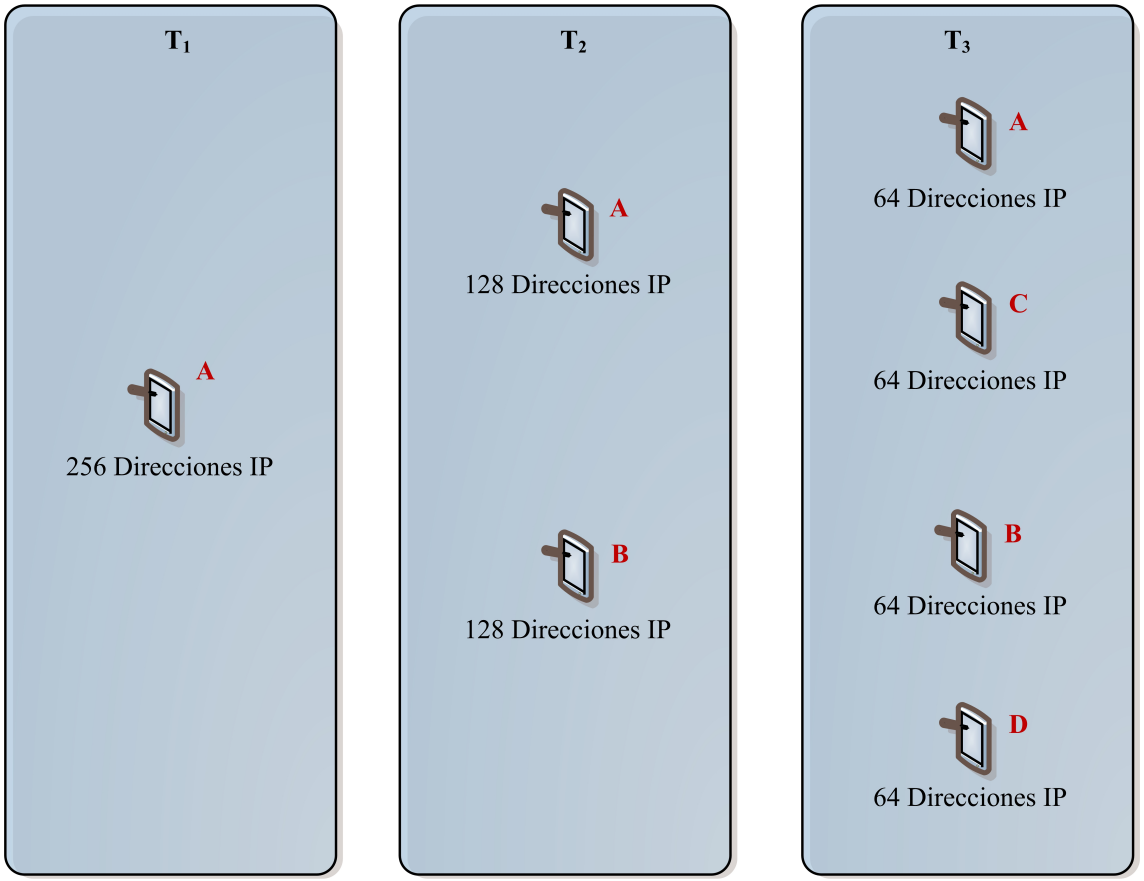



Figura 3.1: Funcionamiento del modelo de división binaria

Inicialmente, el nodo A posee todo el rango de direcciones IP menos una que está asociada a su interfaz de red, por lo que tiene un bloque de 255 direcciones IP libres para entregar a otros nodos.

Cuando el nodo A recibe una solicitud de entrada a la red del nodo B, divide su conjunto de direcciones IP libres en dos mitades, suministrando la mitad para el nodo B y quedándose con la otra mitad. Ahora, el nodo A y el nodo B tienen cada uno 128 direcciones, siendo una dirección IP para su interfaz de red y 127 para otras peticiones.

Usando el mismo mecanismo el nodo A divide su conjunto de direcciones IP libres en dos mitades nuevamente, suministrando la mitad para el nodo C y quedándose con la otra mitad. El mismo procedimiento ocurre con el nodo B que atiende una petición del nodo D.

El mecanismo de división binaria asegura que todos los nodos de la red posean conjuntos disjuntos de direcciones IP, evitando que una misma dirección IP pueda ser usada por dos o más nodos aún cuando se produzca la unión de dos redes ad hoc. Es decir, asegura completamente la unicidad de las direcciones IP.

Tomando como base este modelo de distribución de direcciones, se muestran a continuación todos los escenarios que pueden ocurrir teniendo en cuenta la topología dinámica de las redes móviles ad hoc.

### Inicialización de la Red

El proceso de inicialización de la red consiste en el intento de conexión de un nodo a una red existente, y ésta no existe, el nodo crea su propia red.

Para asociarse a una red un nodo envía mensajes ADDR_REQ (*Address_Request*) solicitando una dirección IP y un bloque de direcciones libres. Si en un número definido de intentos no recibe respuesta alguna, creará una red propia y generará un PID que quedará asociado a la nueva red. De este modo se convertirá en el nodo líder que poseerá todas las direcciones libres de la red. Para su interfaz reservará la primera dirección.

### Entrada de Nodos

Cuando un nodo desea unirse a la red con el objetivo de obtener una dirección IP envía un mensaje *ADDR_REQ* por difusión. Estos primeros mensajes se envían usando la capa MAC, ya que no se dispone aún de una dirección IP para el nuevo nodo. Cualquier nodo perteneciente a la red responde a la petición del nodo cliente enviando un mensaje ADDR_REP (*Address Response*). Este mensaje contiene el mayor bloque *Free_IP_Blocks* de entre los que dispone, ya que el nodo puede poseer más de uno con diferente número de direcciones IP.

El nodo cliente puede recibir más de un mensaje de diferentes nodos y elegirá como nodo servidor al que le haya enviado el mayor *Free_IP_Blocks* enviándole un mensaje *SERVER_POLL*.

Una vez se haya recibido el mensaje *SERVER_POLL* del nodo cliente confirmando la intención de obtener una dirección IP, el nodo servidor divide su *Free_IP_Blocks* por la mitad, suministrando una mitad al nodo cliente y quedándose la otra mitad para atender futuras peticiones.

Esta mitad del bloque es enviada a través del mensaje *IP_ASSIGNED*. El nodo cliente asocia la primera dirección IP de este bloque recibido como su dirección de red y marca al bloque como el *Free_IP_Blocks* que almacena su propia dirección IP.

Para confirmar que la configuración fue realizada con éxito, el nodo cliente envía un mensaje *IP_ASSIGNMENT_OK* al nodo servidor.

A continuación informa de su presencia a cada uno de los nodos de la red enviando un mensaje *NODE_UP* a cada uno de ellos las veces que sean necesarias, hasta recibir las correspondientes respuestas de confirmación *NODE_UP_REPLY* de cada uno. Estos mensajes ya se envían mediante datagramas IP.

El nodo cliente establecerá en sus estructuras de datos internas como *father* al nodo que le entregó el *Free_IP_Blocks*.

La Figura 3.2 muestra el intercambio de mensajes durante el proceso de entrada de un nodo en la red.

Puede darse el caso de pérdidas de mensajes en el proceso de entrada de los nodos. Si el nodo servidor no recibe el *IP_ASSIGNMENT_OK*, envía un mensaje *ping* al nodo cliente para verificar que fue definitivamente configurado. Si recibe respuesta, eso implica que el proceso de autoconfiguración fue realizado con éxito.

Por lo tanto, el mensaje de confirmación se perdió.

Por otro lado, si el nodo cliente no recibe algún mensaje *NODE_UP_REPLY*, comprobará que los nodos siguen aún en la red mediante el envío de mensajes *ping* y, si es así, reenviará el mensaje *NODE_UP* hasta recibir respuesta.

Es posible que un nodo servidor no tenga ninguna dirección IP disponible en su *Free_IP_Blocks* al recibir un mensaje de petición de un nodo entrante. La solución es encaminar la petición hacia los nodos vecinos en la red. Para mantener una distribución

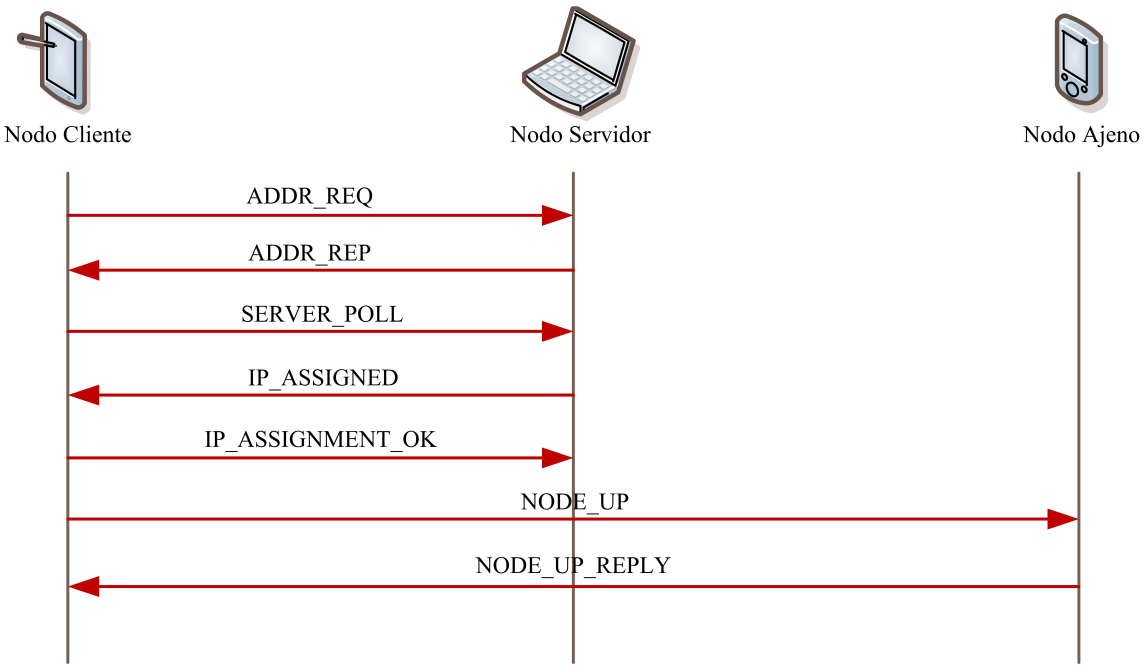



Figura 3.2: Intercambio de mensajes en el proceso de entrada de un nodo en la red

uniforme de las direcciones IP, el nodo servidor mira en su tabla *Allocated_IP_Blocks* para buscar cuál es el nodo que posee mayor número de direcciones libres.

Si ninguno de los nodos posee una dirección IP libre, el nodo servidor responde al cliente enviando un mensaje *DENY* informando que no hay direcciones IP disponibles en ese momento.

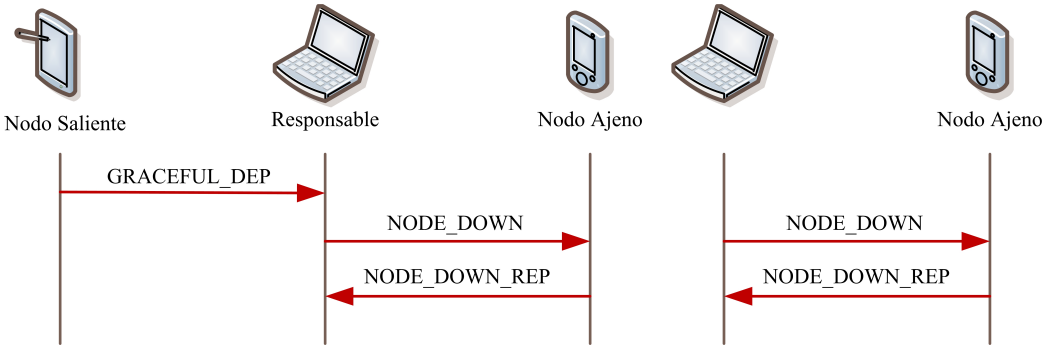
### Salida de Nodos

La salida de nodos en este protocolo se trata de dos maneras diferentes, diferenciando entre salida fácil y abrupta. Una salida abrupta está causada por ejemplo por la desconexión de un nodo, el agotamiento de la batería o simplemente el desplazamiento hacia una zona fuera del rango de cobertura de la red.

- **Salida fácil**: Cuando un nodo desea abandonar la red lo notifica facilitando la tarea de recuperar las direcciones que dejará libres. Para realizar la notificación comprueba si su padre se encuentra en la red. Si su padre responde a mensajes *ping*, entonces le envía un mensaje *GRACEFUL_DEP* y abandona la red.

  Si su padre no se encuentra en la red, el nodo responsable de recoger sus direcciones es el nodo con mayor antigüedad. Por lo tanto, es a él a quien envía el mensaje *GRACEFUL_DEP*. Tanto si es el padre como si es el nodo de mayor antigüedad, el nodo responsable recoge las direcciones IP y actualiza su tabla con los nuevos cambios. Después envía un mensaje *NODE_DOWN* a todos los nodos de la red avisando que un nodo ha abandonado la red. De este modo todos actualizan las tablas. Para confirmarlo responden con el mensaje *NODE_DOWN_REP*. Si algún nodo no responde con este mensaje, se reenvía el mensaje *NODE_DOWN* hasta que todos los nodos tengan constancia del cambio. Si se pierde algún mensaje de salida, se considerará salida abrupta. El proceso se puede observar en la Figura 3.3(a).

- **Salida abrupta**: Si un nodo sale de forma abrupta no tiene ocasión de avisar a ningún otro de su salida. Por tanto, sus *Free_IP_Blocks* se perderían si no se tuviese un mecanismo diseñado para estas situaciones. Los nodos comprueban periódicamente que su nodo padre y sus nodos hijos siguen en la red mediante mensajes *ping*. Si los nodos hijos han abandonado la red, el nodo recupera las direcciones que le asignó. Por otra parte, si es su nodo padre el que ha abandonado la red, el nodo se encargará de sus direcciones sólo si es el nodo más antiguo de la red. Notificará mediante mensajes *NODE_DOWN* la caída del nodo, para que el resto de integrantes de la red mantengan sus tablas actualizadas. El intercambio de mensajes en este proceso puede observarse en la Figura 3.3(b).




(a) Salida Fácil (b) Salida Abrupta

Figura 3.3: Intercambio de mensajes en el proceso de salida de un nodo de la red

### Sincronización

Para que todos los nodos tengan actualizadas las tablas que representan el estado de la red es necesario un proceso de sincronización que permita detectar los cambios producidos en la misma.

Cada nodo dispone de una tabla *Allocated_IP_Block*, que contiene información de todos los nodos de la red: los padres de cada uno, sus direcciones libres y su TS. Tiene que ser igual en todos los nodos, y ante posibles cambios en la red, debe ser actualizada en el menor tiempo posible.

En el caso de entrada de nodos a la red la sincronización se realiza enviando los mensajes *NODE_UP* y *NODE_UP_REPLY*.

Cuando se produce una salida fácil se informa de la salida mediante el mensaje *GRACEFUL_DEP*. El nodo que recogerá sus direcciones IP avisará al resto de nodos mediante el mensaje *NODE_DOWN*. Si se da el caso de que la salida es abrupta, también puede detectarse ya que se dispone de un temporizador de sincronización para que cada nodo realice una comprobación de los nodos de los que es responsable cada cierto tiempo. Cada nodo se responsabiliza de los nodos a los que ha entregado direcciones libres, y de su nodo *father*. En caso de que un nodo no tenga a otro responsable de él, lo será el nodo más antiguo de la red. Si en una de estas comprobaciones mediante mensajes *ping* un nodo comprueba la desaparición de otro, realizará la sincronización notificando al resto de la caída del mismo.

#### Partición y Fusión de Redes

El protocolo soporta la partición y fusión de redes. El principio que rige este funcionamiento es simple: cada red tiene un PID. Cuando un subgrupo de la red la abandona, esto es, se produce una partición, el mecanismo de actualización detectará el cambio como si se tratase de varias salidas abruptas al mismo tiempo. Por lo tanto, la sincronización está resuelta en el caso de división de redes.

Al fusionarse dos redes se dispone de un mecanismo que intenta resolver los conflictos que se puedan dar (direcciones duplicadas). Se basa en el PID y en el campo TS de cada nodo para detectar cuando dos nodos diferentes comparten dirección IP.

## 3.6 Síntesis del Capítulo

El principal objetivo de este capítulo ha sido mostrar la necesidad en las redes móviles ad hoc de un protocolo que realice la configuración de la red de forma dinámica y automática, que utilizará todos los nodos de la red (o sólo una parte de ellos) como si fuesen servidores que gestionan direcciones IP. Se ha visto que hay tres grandes tipos de protocolos de autoconfiguración (protocolos de estado completo, protocolos sin estado y protocolos híbridos) dependiendo del conocimiento del estado de la red por parte de los nodos. Se ha hecho una revisión de los protocolos de autoconfiguración más importantes, mostrándose ésta atendiendo a la clasificación anterior y al orden cronológico. Asimismo, se ha analizado en detalle el protocolo de Mohsin & Prakash: entrada y salida de nodos, cómo funciona la sincronización, las estructuras de datos del protocolo, el funcionamiento general de protocolo, el proceso de inicialización de la red, el comportamiento del protocolo en los procesos de partición y fusión de la red), dado que éste es el precursor inmediato de los protocolos especificados en los capítulos 6 y 7 de esta memoria.

# Capítulo 4

# Encaminamiento en Redes Móviles Ad Hoc

Este capítulo está dedicado a la problemática del encaminamiento en redes móviles ad hoc, otro de los aspectos fundamentales de este tipo de redes. Se comienza viendo la necesidad del diseño de protocolos de encaminamiento específicos para las redes móviles ad hoc dada la naturaleza de las mismas, así como las características o requisitos que deben cumplir para funcionar adecuadamente, comentándose también la imposibilidad de utilizar las soluciones tradicionales. Posteriormente, se muestran diversas clasificaciones de los protocolos de encaminamiento para redes móviles ad hoc (en función de la información de estado que almacenan los nodos de la red, de la estructura y del procedimiento adoptado para el descubrimiento del camino a establecer). Se presta especial atención al protocolo OLSR por estar íntimamente relacionado con los protocolos de autoconfiguración desarrollados en el presente trabajo. El capítulo finaliza con una breve síntesis de lo expuesto en el mismo.

## 4.1 Protocolos de Encaminamiento

En redes móviles ad hoc los protocolos convencionales o bien tendrán un rendimiento muy pobre, o bien serán simplemente inaplicables. Como alternativa se desarrollan protocolos específicos de encaminamiento. Con frecuencia se les denomina de nivel 2.5, ya que es habitual encontrarlos por encima de protocolos de enlace como IEEE 802.11 y por debajo del protocolo de red IP.

El concepto de encaminamiento básicamente comprende dos actividades. En primer lugar, determinar los caminos óptimos y, en segundo lugar, transferir los grupos de paquetes de información a través de la red. Los algoritmos utilizan varias métricas para calcular el mejor camino para que los paquetes lleguen a su destino. Estas métricas son medidas estándar como podría ser el número de saltos que son usados por el algoritmo para determinar el camino óptimo. El proceso para determinar el camino inicializa y mantiene tablas de encaminamiento que contienen la información total de cada ruta. La información que se almacena para cada ruta varía de un algoritmo a otro.

Las redes móviles ad hoc se construyen de forma dinámica cuando un conjunto de nodos crean rutas entre sí para conseguir la conectividad entre ellos. Los nodos de la red móvil ad hoc pueden actuar como origen o destino de una comunicación, pero también como encaminadores cuando una relación entre nodos no se puede realizar directamente por motivos de alcance. De esta forma se crean comunicaciones multisalto. Un protocolo de encaminamiento de una red móvil ad hoc necesita proveer un mecanismo que mantenga

las rutas hacia los destinos frente al movimiento de los nodos que puede provocar que las rutas se destruyan, y sea necesario encontrar una ruta alternativa para mantener la comunicación entre los nodos.

El objetivo de un protocolo de encaminamiento para redes móviles es conseguir el envío de un mensaje de un nodo a otro sin existir un enlace directo. La mayoría de protocolos de encaminamiento para redes móviles ad hoc provienen de adaptaciones realizadas sobre protocolos de redes fijas, siendo su principal problema la cantidad de fallos que se producen en la comunicación debido a la movilidad de los nodos.

Los protocolos de encaminamiento para redes móviles ad hoc deben satisfacer básicamente los siguientes criterios [BR05]:

- **Señalización mínima**: La reducción de los mensajes de control ayuda a conservar la capacidad de las baterías y la comunicación de los nodos.
- **Mantenimiento dinámico de topología**: El algoritmo deberá ser capaz de localizar una nueva ruta rápidamente cuando se rompe un enlace.
- **Libre de bucles**: Se pretende evitar el problema de tener paquetes circulando perdidos por la red.
- **Capacidad multisalto**: Debe asegurarse el reenvío de paquetes a través de los nodos de la red dado que habitualmente el destino no se encuentra dentro del alcance de la fuente.
- **Tiempo de procesamiento mínimo**: Se requieren algoritmos con cálculos computacionales que no sean excesivamente complejos para disminuir el tiempo de procesamiento y alargar de esta forma el tiempo de vida de la batería.

Además, debe admitir diversos modos de operación [MC05]:

- **Distribuido**: Propiedad esencial de las redes móviles ad hoc.
- **Inactivo**: Los protocolos de encaminamiento deberán estar preparados para afrontar aquellos períodos de tiempo en los cuales los nodos frenan su actividad y permanecen inactivos para ahorrar energía.
- **Bajo demanda**: La adaptación del encaminamiento a los patrones de tráfico particulares de cada situación hace posible reducir el gasto de ancho de banda y energía, aunque se amplía el tiempo de obtención de la ruta.
- **Soporte de enlaces unidireccionales**: Los protocolos de encaminamiento en muchas ocasiones han sido diseñados y funcionan correctamente sólo con enlaces bidireccionales y esto no debería ser así, porque en la práctica pueden existir enlaces unidireccionales que sean clave para el intercambio de información en redes móviles ad hoc.

Se han diseñado numerosos protocolos de encaminamiento para redes móviles ad hoc atendiendo a estos criterios.

La finalidad del *Mobile Ad Hoc Networks Work Group* (Manet WG) [Man] del IETF es estandarizar la funcionalidad de un protocolo de encaminamiento IP para aplicaciones de encaminamiento inalámbrico dentro de topologías tanto estáticas como dinámicas como consecuencia de la movilidad de los nodos u otros factores. Los enfoques están destinados a ser relativamente generales pues deben ser adecuados en múltiples entornos inalámbricos

y hardware y dirigidos a escenarios donde las redes móviles ad hoc estén desplegadas en la frontera de una infraestructura IP. Las infraestructuras *mesh* híbridas (por ejemplo, una mezcla de *routers* móviles y fijos) deberían ser soportados por las especificaciones de las redes móviles ad hoc.

## 4.2 Clasificación de los Protocolos de Encaminamiento

Desde que se empezaron a estudiar las redes móviles ad hoc se han propuesto diversas clasificaciones de los protocolos de encaminamiento que se resumen en [JG07].

*En función de la información de estado que almacenan los nodos de la red* los protocolos pueden ser clasificados en *protocolos basados en la topología* y *protocolos basados en el destino*. En los primeros cada nodo toma decisiones basándose en una completa información de la topología de la red. Los segundos son protocolos que manejan vectores de distancias, en los que cada nodo intercambia con sus vecinos las distancias que conoce a otros nodos.

Otra clasificación propone dividir los protocolos en *función de la estructura*, diferenciándose varios niveles. La Figura 4.1 presenta esta clasificación, especificando algunos de los protocolos representativos de cada categoría.

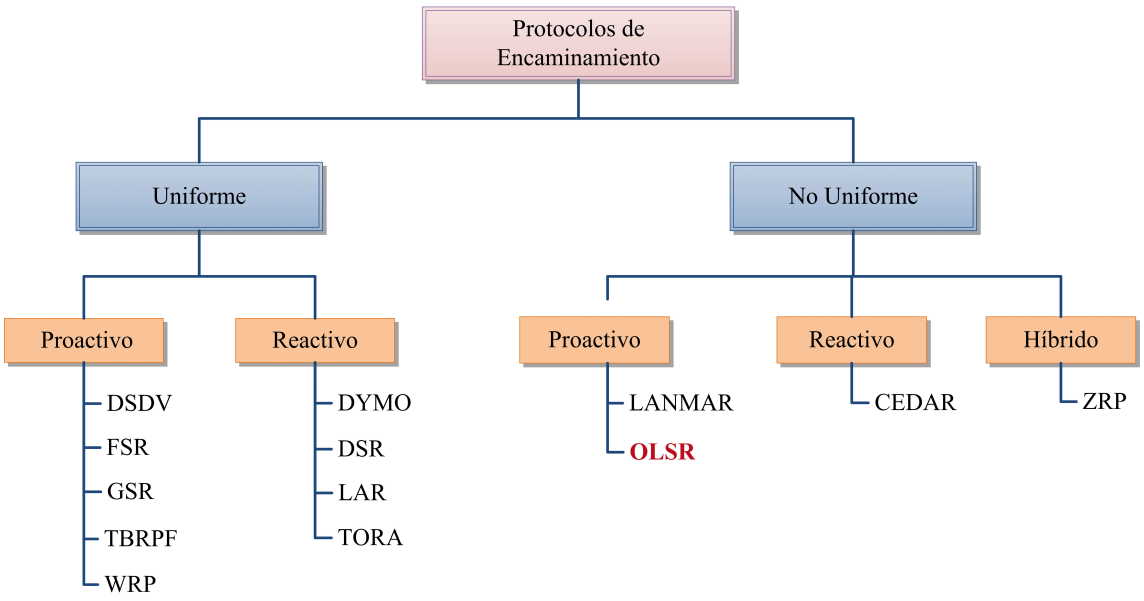



Figura 4.1: Taxonomía de protocolos de encaminamiento en redes móviles ad hoc

*El primer nivel se refiere a la homogeneidad o heterogeneidad de las funciones de los nodos en el encaminamiento*, distinguiéndose dos tipos:

- **Protocolos uniformes o de estructura plana**: Ningún nodo de la red realiza un papel distinto al de los demás, todos ellos envían y responden a los mensajes de control del mismo modo.
- **Protocolos no uniformes**: Típicos de estructuras jerárquicas en las que algunos nodos desarrollan papeles especiales e incluso pueden dotarse de capacidades particulares en términos de cómputo, energía o almacenamiento, entre otras. Esto les

permite soportar algoritmos más complejos, reducir la sobrecarga debida a la comunicación y ofrecer la posibilidad de balanceo de carga mientras mantienen sus características, incluso ante incrementos del número de nodos en la red. Por el contrario, generan cierto coste de mantenimiento de la estructura y necesitan en muchos casos la disponibilidad de nodos heterogéneos.

*Dentro de estas dos categorías anteriores, los protocolos presentan una nueva peculiaridad relativa al procedimiento adoptado para el descubrimiento del camino a establecer y su mantenimiento.* Esta clasificación, sin duda, es la más difundida, surgiendo los siguientes tipos de protocolos:

- **Protocolos proactivos**: En este tipo de encaminamiento cada nodo mantiene información de cómo llegar a cualquier otro nodo de la red e intercambia esta información con todos sus vecinos. La información de encaminamiento es normalmente almacenada en un número diferente de tablas. Periódicamente se actualizan las tablas si la topología de red cambia. La diferencia entre los protocolos de este tipo se encuentra en la forma de actualizar y detectar la información de encaminamiento y el tipo de información que se almacena en cada tabla. La ventaja que aportan estos protocolos es la baja latencia ya que las rutas están siempre disponibles. Sin embargo, esto conlleva un consumo de energía muy alto en los nodos y se puede producir una sobrecarga de mensajes en la red debido a la inundación periódica de mensajes. Seguidamente se enumeran los protocolos de encaminamiento proactivos más representativos:

  - *Destination-Sequenced Distance-Vector* (DSDV) [PB94].
  - *Wireless Routing Protocol* (WRP) [MGLA95].
  - *Global State Routing* (GSR) [CG98].
  - *Fisheye State Routing Protocol* (FSR) [GHP02].
  - OLSR [CP03].
  - *Topology Broadcast Reverse Path Forwarding* (TBRPF) [OTL04].

  En general, estos protocolos tratan de evitar bucles en las rutas, consumo excesivo de memoria y reducción del tamaño de los paquetes que contienen la información de las tablas de encaminamiento.

  Dentro de los protocolos proactivos se pueden distinguir dos subtipos de protocolos según su comportamiento: *los conducidos por eventos* (*event-driven*), que envían paquetes con información sobre las rutas sólo cuando éstas sufren algún cambio, y *los que refrescan la información periódicamente* (*regular updated*). El protocolo OLSR, que ha sido utilizado para este trabajo y que será analizado en detalle en el siguiente apartado, entra dentro de la segunda categoría.

- **Protocolos reactivos**: Estos protocolos tratan de reducir la sobrecarga que producen los protocolos proactivos. Para ello proponen que los nodos de la red móvil ad hoc, cuando no tienen una ruta a un destino, la calculen sólo cuando es necesaria, es decir, cuando el nodo tenga que comenzar un intercambio de paquetes con el destino. El descubrimiento de una ruta normalmente se realiza por inundación de mensajes de solicitud por toda la red. Estos protocolos conllevan una alta latencia, provocada por el descubrimiento de rutas. Sin embargo, la sobrecarga de mensajes por la red se reduce. Seguidamente se enumeran los protocolos de encaminamiento reactivos más representativos:

  - *Lightweight Mobile Routing* (LMR) [CE95].
  - *Routing On-Demand Acyclic Multi-Path* (ROAM) [RGLA99].
  - *Location Arder Routing* (LAR) [KV00].
  - *Temporally-Ordered Routing Algorithm* (TORA) [PC01].
  - AODV [PBRD03].
  - *Dynamic Source Routing* (DSR) [JHM07].
  - *Dynamic Manet On-Demand* (DYMO) [CP10].

  La mayoría de ellos tienen el mismo coste de encaminamiento en el peor escenario posible ya que casi todos siguen la misma filosofía para el descubrimiento de rutas.

- **Protocolos híbridos**: Combinando los protocolos proactivos y reactivos nacen los protocolos híbridos que pretenden minimizar los inconvenientes de ambos. La idea de estos protocolos es que los nodos de la red trabajen de forma proactiva con los nodos más cercanos y de forma reactiva con el resto de nodos. La parte reactiva controla la sobrecarga y el consumo de memoria al calcular las rutas sólo cuando son necesarias. En contraste, la parte proactiva necesita actualizar periódicamente la información almacenada y mantiene rutas que quizás nunca serán utilizadas, añadiendo una innecesaria sobrecarga. El caso más conocido de protocolo híbrido es *Zone Routing Protocol* (ZRP) [HPP02].

El Grupo de Trabajo Manet WG tiene previsto desarrollar dos especificaciones de protocolo de encaminamiento estándar, denominadas *Reactive Manet Protocol* (RMP) y *Proactive Manet Protocol* (PMP), si bien también puede decidir un enfoque mixto. Soportará IPv4 e IPv6, requisitos de seguridad y otros aspectos, y prestará atención especial al protocolo OSPF-MANET que viene desarrollando el *Open Shortest Path First IGP Work Group* (OSPF WG) [OSP]. El OSPF WG desarrolla extensiones del protocolo *Open Shortest Path First* (OSPF) para diferentes escenarios, siendo OSPF-MANET la extensión de OSPF a redes móviles ad hoc.

## 4.3 Protocolo OLSR

El protocolo de encaminamiento *Optimized Link State Protocol* (OLSR) pertenece al grupo de los protocolos para redes móviles ad hoc que están definidos como RFC. OLSR se especifica en la RFC 3626 [CP03], siendo una optimización del clásico protocolo de estado de enlace u OSPF [Moy98], pero adaptado a redes móviles ad hoc.

Al tratarse de un protocolo proactivo, la información sobre las rutas hacia todos los nodos se mantiene siempre actualizada, para que esté disponible en caso de que sea necesaria. Como se ha indicado anteriormente, OLSR es *regular updated*, esto es, cada cierto tiempo paquetes de información sobre las rutas son transmitidos, aunque no se hayan detectado cambios.

La principal aportación de OLSR que lo diferencia de otros protocolos similares son las optimizaciones que se realizan para que la sobrecarga producida por las actualizaciones periódicas sea mínima. El modo en que se hace llegar la información sobre rutas a toda la red es mediante *inundación controlada*: designando a ciertos nodos encargados de enviarse entre ellos la información, haciéndola llegar posteriormente al resto de la red, y comprobando que no se envían datos duplicados.

Debido a las optimizaciones que aplica, obtiene buenos resultados en redes grandes y densas. Sus optimizaciones se notan en el rendimiento cuanto más grandes sean las redes,

sobre todo si el tráfico es esporádico entre pares de nodos que varíen irregularmente, en lugar de comunicaciones regulares entre nodos concretos.

### 4.3.1 Funcionamiento del Protocolo

Lo primero que hace un nodo al iniciarse es detectar con qué otros nodos tiene conexión a nivel de enlace. Para ello periódicamente se emiten mensajes *Hello*. Estos mensajes no se retransmiten por los nodos que los reciben, ya que su finalidad es que los nodos se den a conocer a sus vecinos de un salto, es decir, nodos con los que existe conectividad a nivel de enlace. Otra funcionalidad de los mensajes *Hello*, aparte de dar a conocer al propio nodo, es anunciar los vecinos ya conocidos del nodo emisor. De esta forma, un nodo que escucha estos mensajes no sólo descubre a sus vecinos de un salto, sino que además adquiere conocimiento de sus vecinos de dos saltos de distancia.

La clave del protocolo está en los Multipuntos de Retransmisión, abreviadamente MPR. Una vez que un nodo conoce el conjunto de sus vecinos de dos saltos, elige de entre sus vecinos de un salto un grupo de nodos MPR que retransmitan sus mensajes, de forma que le proporcionen acceso hacia todos los vecinos de dos saltos; de esta forma abrirán la ruta hacia cualquier nodo de la red. Los nodos elegidos como MPR son notificados de su condición, manteniendo información sobre quiénes le han elegido como MPR (denominados selectores MPR) en una estructura llamada conjunto selector MPR.

Uno de los cometidos de los MPR es retransmitir los mensajes de difusión generados por alguno de los nodos en su conjunto selector MPR. De este modo, los mensajes llegan a toda la red, pero intentando que la saturación sea mínima.

La otra tarea que debe realizar un nodo que ha sido seleccionado como MPR es generar y retransmitir mensajes *Topology Control* (TC), que dan a conocer al resto de la red los nodos de los que el emisor tiene constancia. Los mensajes TC se generan de forma periódica y contienen una lista con las direcciones de los nodos del conjunto selector MPR, nodos que han elegido al emisor del mensaje como MPR (a diferencia de otros protocolos que anunciarían a cualquier nodo cercano). Con esto se consigue que la información sobre la topología generada sea mínima y que su diseminación por la red se haga de forma controlada.

En otras palabras, los MPR tienen también la función de anunciar información sobre la topología de la red mediante inundación controlada, para que todos los participantes conozcan la ruta hacia el resto de la red. Como se ha indicado anteriormente, la inundación se realiza mediante mensajes TC, generados por nodos que han sido seleccionados como MPR, y que se reenvían de forma eficiente entre los MPR en lugar de hacerse mediante difusión masiva.

También es importante conocer las estructuras de datos internas que maneja OLSR. En particular, la más relevante es la *tabla de rutas*. La información que se tiene de cada nodo de la red en la tabla de rutas es una entrada con los siguientes campos:

***R_dest_addr*** ***R_next_addr*** ***R_dist*** ***R_iface_addr***

Cada entrada significa que el nodo identificado por la dirección *R_dest_addr* está a una distancia estimada de *R_dist* saltos, que el vecino con la dirección de interfaz *R_next_addr* es el siguiente salto en la ruta hacia *R_dest_addr* y que este vecino es alcanzable a través de la interfaz local con la dirección *R_iface_addr*.

OLSR tiene en consideración la posibilidad de que un nodo tenga más de una interfaz de red participando al mismo tiempo en la red. Cada dirección de interfaz está asociada con una dirección principal, única para cada nodo. Esta dirección principal será la misma

que la dirección de interfaz en caso de que el nodo tenga una única interfaz utilizando OLSR.

### 4.3.2 Formato del Paquete OLSR

Las comunicaciones en OLSR se realizan usando un formato de paquete común para toda la información relacionada con el protocolo. De este modo, se facilitan futuras ampliaciones del protocolo sin romper la compatibilidad con versiones anteriores. Además, esto también facilita agrupar diferentes tipos de información dentro de una misma transmisión.

Los paquetes se encapsulan dentro de datagramas *User Datagram Protocol* (UDP) para su transmisión por la red.

Cada paquete encapsula a su vez uno o varios mensajes. Los mensajes comparten un formato de cabecera común, lo que permite que un nodo sea capaz de aceptar y retransmitir (si procede) mensajes de tipo desconocido.

Los mensajes pueden ser transmitidos por inundación a la red en su totalidad o la transmisión puede ser limitada a nodos dentro de un determinado diámetro -refiriéndose a número de saltos- desde el emisor del mensaje. Por lo tanto, transmitir un mensaje al vecindario de un nodo es un caso especial de transmisión por inundación. Cuando se transmiten mensajes de control, las retransmisiones duplicadas son eliminadas de forma local, ya que cada nodo almacena información sobre los mensajes de control que ya ha transmitido anteriormente.

Los paquetes en OLSR usan el puerto UDP 698, asignado por el *Internet Assigned Numbers Authority* (IANA).

Los campos de cualquier paquete OLSR (omitiendo las cabeceras IP y UDP) se indican en la Figura 4.2.

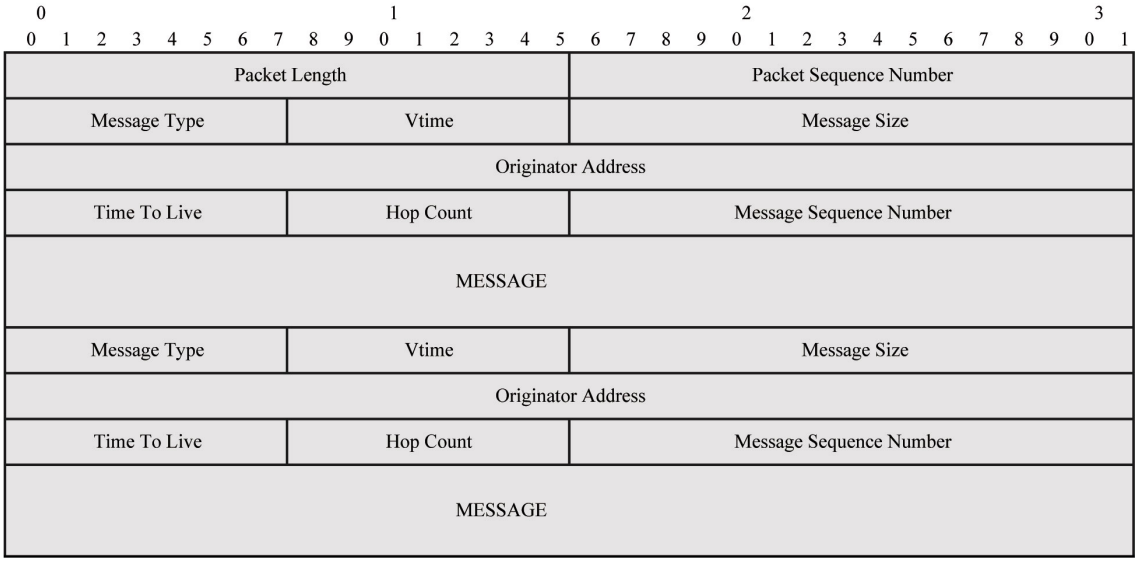



Figura 4.2: Formato del paquete OLSR

#### Cabecera del paquete OLSR

La Figura 4.3 muestra los campos de la cabecera del paquete OLSR.

Como se observa, estos campos son los siguientes:

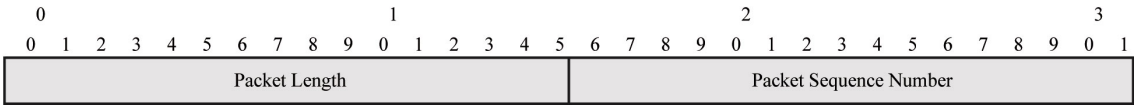



Figura 4.3: Cabecera del paquete OLSR

- **Packet Length** [16 bits]: Este campo define el tamaño del paquete OLSR en bytes.
- **Packet Sequence Number** [16 bits]: Este campo se utiliza para definir el número de secuencia del paquete. Debe ser incrementado en uno cada vez que un nuevo paquete OLSR es transmitido.

### Cabecera de los mensajes de OLSR

La Figura 4.4 muestra los campos de la cabecera de cada mensaje.

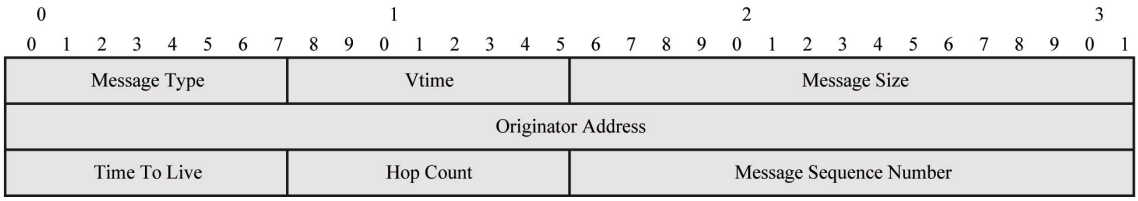



Figura 4.4: Cabecera de los mensajes OLSR

Como se observa, estos campos son los siguientes:

- **Message Type** [8 bits]: Este campo indica qué tipo de mensaje se encuentra en el campo MESSAGE. Los tipos en el rango 0-127 son reservados.
- **Vtime** [8 bits]: Campo que indica el tiempo de validez de la información contenida en el mensaje.
- **Message Size** [16 bits]: Tamaño del mensaje en bytes, incluyendo la cabecera y el campo MESSAGE.
- **Originator Address** [32 bits]: Indica la dirección principal del nodo que originalmente generó el mensaje. No debe confundirse este campo con la dirección origen de la cabecera IP, que se modifica cada vez que el paquete OLSR se retransmite por un nodo intermedio.
- **Time To Live** [8 bits]: Contiene el número máximo de saltos que el mensaje será transmitido. Antes de retransmitir un mensaje, al valor de este campo se le debe restar uno.
- **Hop Count** [8 bits]: Número de saltos que el mensaje ha dado. Se inicializa a 0 y se incrementa en 1 en cada retransmisión.
- **Message Sequence Number** [16 bits]: El nodo que genera un mensaje le asigna un número de secuencia único que sirve de identificador del mensaje. El número de secuencia se incrementa en 1 cada vez que el nodo origina un mensaje.

### 4.3.3 Mensaje MID

Si el nodo tiene más de una interfaz se anuncia esta interfaz adicional periódicamente a los otros nodos utilizando mensajes *Multiple Interface Declaration* (MID). El formato del mensaje se indica en la Figura 4.5.

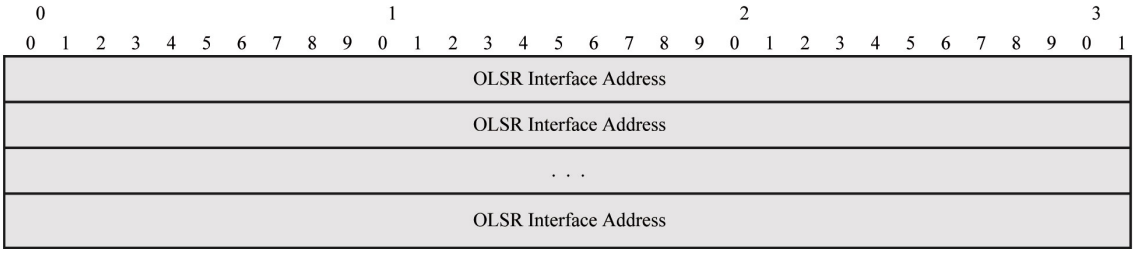



Figura 4.5: Formato del mensaje MID

### 4.3.4 Mensaje Hello

El protocolo OLSR utiliza el intercambio periódico de mensajes *Hello* para descubrir a los nodos vecinos y el estado de la red a nivel de enlace. Como la dirección principal del nodo está incluida en la dirección origen del encabezado del mensaje sólo las direcciones de las interfaces adicionales tienen que ser anunciadas. A partir de esta información se construye el *Multiple Interface Association Information Base* en el nodo receptor.

Los mensajes *Hello* se envían como datos dentro del paquete general OLSR descrito anteriormente, configurando el campo *Message Type* con el valor *Hello_Message* y el campo TTL con el valor 1.

#### Formato del mensaje Hello

Los mensajes *Hello* siguen un formato similar al del paquete general, de modo que puedan incluir información para la detección del estado de los enlaces de la red, para transmitir señales para la detección de nodos vecinos, para selección de nodos MPR y para tener en cuenta futuras ampliaciones. El formato del mensaje se indica en la Figura 4.6.

Los campos del mensaje *Hello* son los siguientes:

- **Reserved** [16 bits]: Reservado, se establece con el valor 0x0000.
- **HTime** [8 bits]: Este campo especifica el intervalo de emisión de mensajes *Hello* usado por el nodo (esto es, el tiempo hasta el envío del próximo mensaje *Hello*).
- **Willingness** [8 bits]: Indica la disponibilidad que tiene un nodo para reenviar tráfico a otros nodos. Si la disponibilidad de un nodo se define como *Will_Never*, éste nunca será elegido como nodo MPR.
- **Link Code** [8 bits]: Este campo especifica el tipo de enlace que el nodo emisor tiene con los vecinos en su lista. OLSR requiere como mínimo los siguientes tres tipos de enlaces:

  - **Asym_Link**: Indica que los enlaces entre el nodo emisor y sus vecinos son de tipo asimétrico (es decir, solamente es posible escuchar al vecino, pero no es posible establecer un enlace bidireccional con éste).
  - **Sym_Link**: Indica que el enlace entre el emisor y sus vecinos son simétricos (existe un enlace bidireccional).
  - **MPR_Link**: Indica que los nodos definidos en la lista han sido seleccionados por el emisor como MPR.

- **Reserved** [8 bits]: Este campo se reserva para uso futuro. Debe ser puesto a 0x00.
- **Link Message Size** [16 bits]: Este campo define el tamaño del mensaje de enlace, el cual se mide desde el inicio del campo *Link Code* hasta el siguiente campo *Link Code*. Si no existe otro campo *Link Code* dentro del mensaje *Hello*, entonces el valor del campo *Link Code* se mide hasta el fin del mensaje *Hello*.
- **Neighbor Interface Address** [32 bits]: Este campo define la lista de vecinos que se han etiquetado con un *Link Code* en particular.

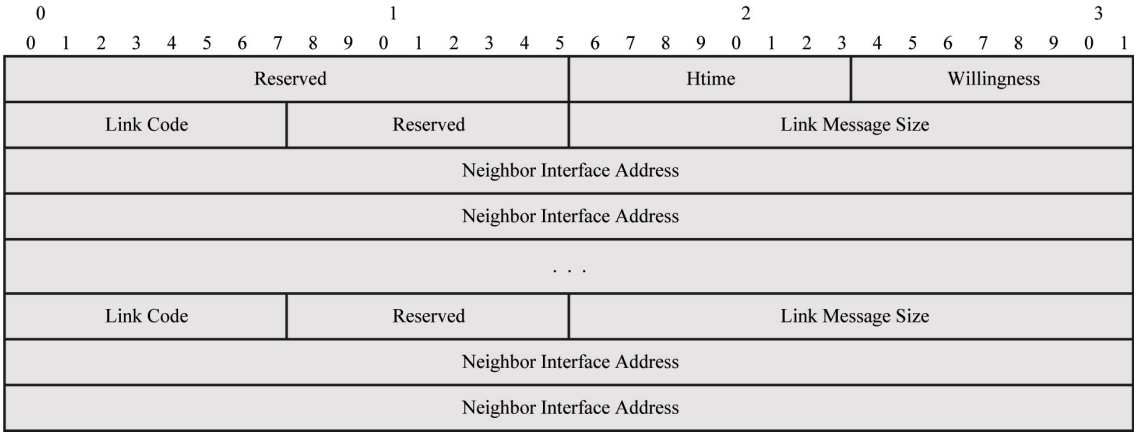



Figura 4.6: Formato del mensaje Hello

#### Procesamiento del mensaje Hello

Los nodos procesan los mensajes *Hello* recibidos para la detección de conexiones a nivel de enlace, detección de vecinos y selección de MPR.

### 4.3.5 Descubrimiento de Vecinos

#### Detección de conexiones a nivel de enlace

Cada nodo almacena información sobre sus conexiones a nivel de enlace con otros nodos en una estructura llamada *Link Set*, entendiendo por conexiones las interfaces de red que utilicen OLSR.

El mecanismo usado para esta detección es el intercambio periódico de mensajes *Hello*. Para considerar una conexión válida se debe comprobar que existe comunicación (recepción de *Hello*) en ambos sentidos.

Cada nodo vecino tiene asociado un estado en relación a la conexión: simétrico o asimétrico. El primero indica que la comunicación en ambos sentidos ha sido confirmada y el segundo se usa para indicar que se han recibido mensajes *Hello* generados por el nodo vecino, pero no se ha confirmado aún que el nodo vecino es capaz de recibir los *Hello* generados localmente. La confirmación de que un nodo vecino es capaz de recibir los mensajes *Hello* emitidos se tiene al encontrar la dirección propia en los *Hello* del vecino.

#### Detección de vecinos

Cada nodo mantiene un conjunto de tuplas de vecinos (*neighbor tuples*) basadas en la información sobre las conexiones almacenadas en el *Link Set*.

Cada tupla consta de los datos:

***N_neighbor_main_addr*** ***N_status*** ***N_willingness***

donde:

- **N_neighbor_main_addr**: Dirección principal del nodo vecino.
- **N_status**: Estado de la conexión (simétrica o asimétrica).
- **N_willingness**: Entero entre 0 y 7 que representa la disposición o intención del vecino de retransmitir tráfico de otros nodos.

Aparte del conjunto de vecinos inmediatos o de un salto, con los que se tiene conexión a nivel de enlace, cada nodo almacena información sobre el conjunto de nodos de dos saltos de distancia en la estructura *2-hop Neighbor Set*. Para ello, por cada nodo vecino de un salto, se almacena el conjunto de sus vecinos de un salto, ya que están anunciados en los mensajes *Hello* que se reciben periódicamente.

### 4.3.6 Multipuntos de Retransmisión

Los *Multipoint Relay* (MPR) se usan para inundar con mensajes de control procedentes de un nodo a toda la red minimizando las retransmisiones. Por lo tanto, el concepto de MPR se considera una optimización del mecanismo habitual de inundación de mensajes en una red.

Como se ilustra en la Figura 4.7, cada nodo en la red elige, independientemente de los demás, su propio conjunto de MPR de entre sus vecinos de un salto con conexión simétrica. El conjunto de nodos seleccionados se conoce como el conjunto MPR de ese nodo. Los vecinos del nodo N que no están dentro del grupo MPR, reciben y procesan la información de los mensajes de difusión, pero no retransmiten la información procedente del nodo N.

La sobrecarga de tráfico de control generada por el protocolo de encaminamiento es directamente proporcional al tamaño del conjunto de nodos MPR en la red. A su vez, los nodos MPR mantienen información sobre el conjunto de vecinos a un salto que lo han seleccionado como MPR; este conjunto se conoce como conjunto selector de MPR de un nodo. Esta información se adquiere de los mensajes *Hello* recibidos de los vecinos a un salto.

Aunque la inundación de mensajes pura es más confiable y robusta, ésta consume una gran cantidad de ancho de banda. El empleo de nodos MPR proporciona resultados igualmente buenos, con mucho menos tráfico de control.

En la Figura 4.8 se ilustra una comparación, en términos de retransmisiones, para hacer llegar un mensaje de difusión a 3 saltos en la red.

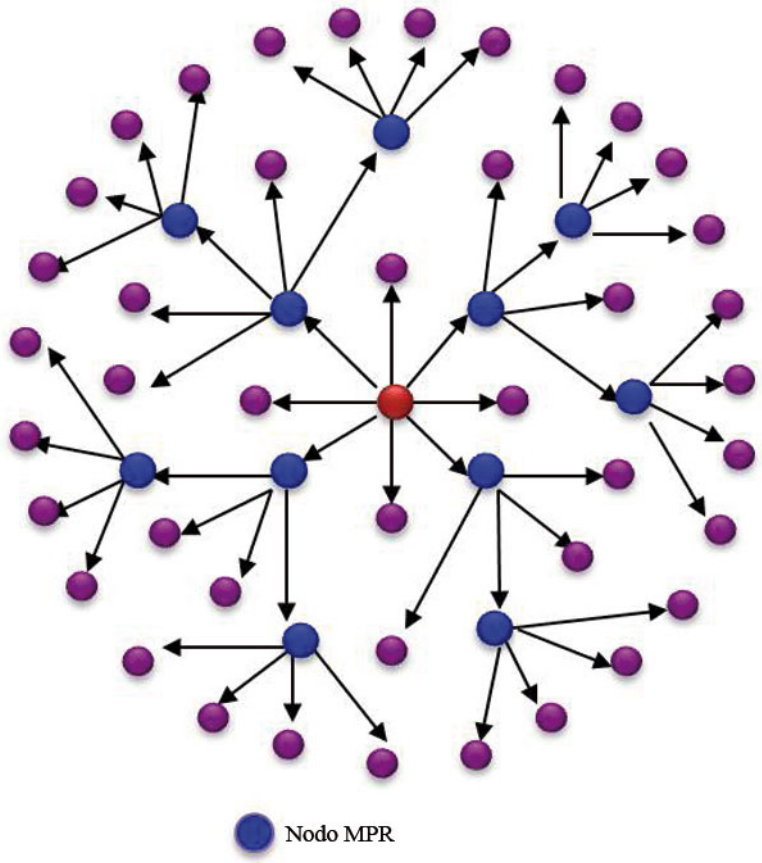
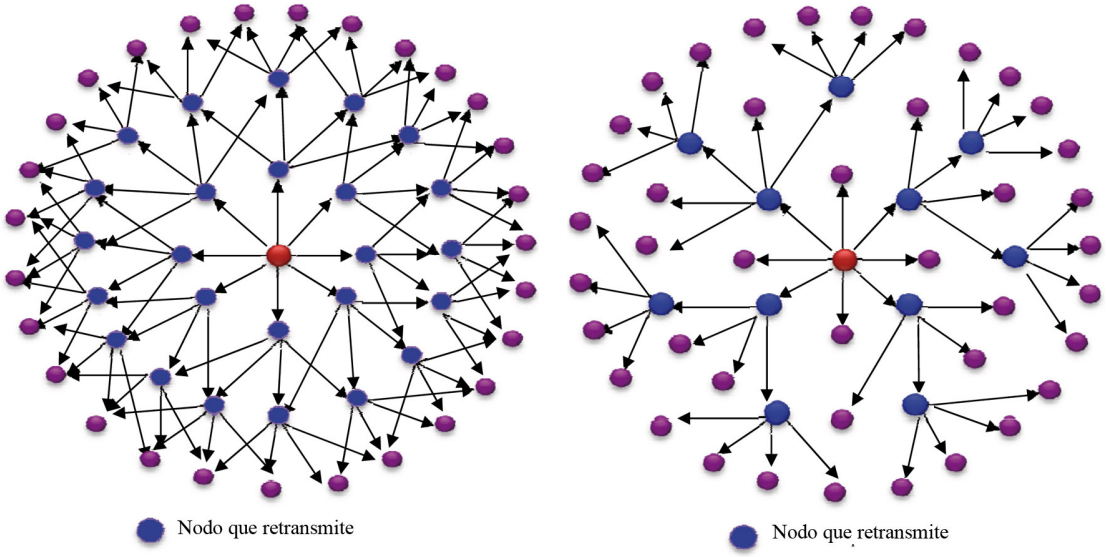



Figura 4.7: Proceso de selección de nodos MPR




(a) Difusión pura (b) Difusión usando nodos MPR

Figura 4.8: Diferencia entre la difusión pura y la difusión usando nodos MPR

#### Selección de MPR

Cuando se ha elegido a un vecino como MPR se anuncia en los mensajes *Hello* colocando el valor *MPR_Neigh* en lugar de *Sym_Neigh* en el campo *Link Type* anterior a la dirección del vecino elegido como MPR.

El conjunto MPR es calculado por cada nodo de forma que, a través de los vecinos elegidos, el nodo sea capaz de alcanzar a todos los vecinos de dos saltos. Más concretamente, a los vecinos de dos saltos exactos, por lo que no se está contando a los vecinos de un salto.

Aunque el conjunto MPR no debe ser mínimo para garantizar el correcto funcionamiento, cuanto menor sea éste menor sobrecarga se produce.

Como se ha indicado anteriormente, cada nodo almacena además en el conjunto selector MPR el conjunto de nodos que le han elegido como MPR. Son detectados al procesar los mensajes *Hello* recibidos.

### 4.3.7 Descubrimiento de la Topología en OLSR

#### Funcionamiento

La detección de conexiones y vecinos proporciona a cada nodo una lista de nodos con los que comunicarse directamente y, haciendo uso de nodos MPR, un mecanismo para inundaciones optimizado. Basándose en esto se genera información sobre la topología y se distribuye por la red.

Los mensajes *Topology Control* (TC) son generados por los nodos que han sido elegidos como MPR por algún vecino suyo. Sirven para anunciar un conjunto de enlaces entre el emisor y otros nodos que, por lo general, será su conjunto selector MPR, es decir, los vecinos que han elegido al nodo emisor como MPR. Estos mensajes TC se emiten a toda la red por inundación.

Debido a limitaciones de tamaño de los mensajes en la red, la lista de direcciones anunciadas puede ser parcial en cada mensaje TC. Sin embargo, al unir todos los mensajes TC emitidos deben encontrarse todas las direcciones del conjunto selector MPR.

#### Formato de los Mensajes TC

Los mensajes TC tienen el formato mostrado en la Figura 4.9.

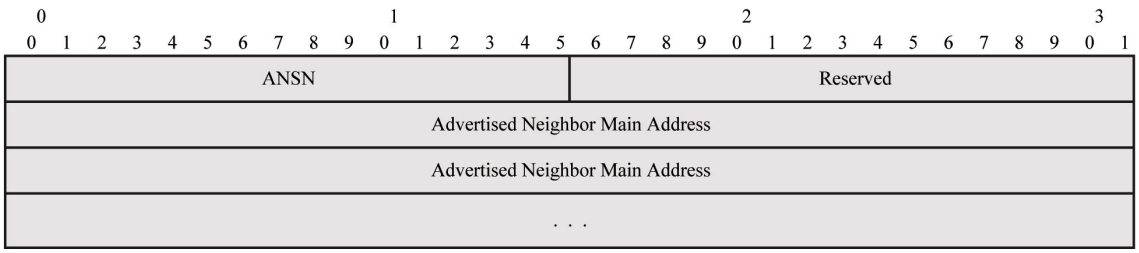



Figura 4.9: Formato del mensaje TC

Este mensaje se envía como datos dentro del paquete OLSR, configurando el campo *Message Type* con el valor *TC_Message* y el campo TTL con el valor 255 (el máximo).

Como se observa, estos campos son los siguientes:

- Advertised Neighbor Sequence Number (ANSN) [16 bits]: Número de secuencia que se asocia con el conjunto de vecinos que se anuncia en este mensaje. Cada vez que un nodo detecta cambios en el conjunto de sus vecinos se incrementa este número. Sirve para que los nodos que reciben este mensaje sepan si se trata de información más reciente que la que puedan tener en ese instante.
- **Reserved** [16 bits]: Este campo está reservado. Se le da el valor 0x0000.
- **Advertised Neighbor Main Address** [32 bits]: Campo que contiene la dirección principal de un nodo vecino.

### 4.3.8 Cálculo de las Tablas de Rutas

Cada nodo mantiene actualizada una tabla con rutas hacia el resto de nodos de la red. Esta tabla se basa en la información obtenida sobre los nodos vecinos y los mensajes de control TC.

El formato de las entradas en esta tabla es el siguiente:

a. ***R_dest_addr*** ***R_next_addr*** ***R_dist*** ***R_iface_addr***

b. ***R_dest_addr*** ***R_next_addr*** ***R_dist*** ***R_iface_addr***

c. ...

Cada entrada significa que el nodo con la dirección *R_dest_addr* se encuentra a una distancia de *R_dist* saltos del nodo local, que el nodo vecino con la dirección de interfaz de red *R_next_addr* es el siguiente salto en la ruta hacia *R_dest_addr* y que este nodo es alcanzable desde la interfaz local con la dirección *R_iface_addr*. Se mantiene una entrada por cada nodo de la red para el que se tiene ruta conocida.

Los nodos con una ruta desconocida no se incluyen. La actualización de la tabla se realiza en caso de aparición o desaparición de un nodo, ya sea vecino inmediato, vecino de dos saltos de distancia, o cualquier otro nodo conocido a través de los mensajes de control TC. También se actualiza la tabla cuando cambia información sobre las interfaces múltiples que puedan estar asociadas a los nodos. Esta actualización de la tabla es un proceso interno, que no desencadena el envío de ningún mensaje.

## 4.4 Síntesis del Capítulo

El principal objetivo de este capítulo ha sido mostrar que el envío de un mensaje de un nodo a otro sin existir un enlace directo constituye la finalidad de los protocolos de encaminamiento para redes móviles ad hoc. Asimismo, se han presentado diversas clasificaciones de los mismos siendo la más relevante aquella que los agrupa según el procedimiento adoptado para el descubrimiento del camino a establecer y a su mantenimiento y que divide a los protocolos de encaminamiento en tres clases: proactivos (protocolos en los que cada nodo mantiene información de cómo llegar a cualquier otro nodo de la red e intercambia esta información con todos sus vecinos) con baja latencia y alta sobrecarga, reactivos (protocolos en los que un nodo sólo calcula la ruta a un destino cuando es necesario un intercambio de paquetes con el mismo) con alta latencia y baja sobrecarga e híbridos (protocolos que combinan aspectos de los dos anteriores, siendo proactivos a nivel local y reactivos a nivel global) que minimizan los inconvenientes de ambos pero a costa de aumentar la complejidad. Asimismo, se ha analizado en detalle el protocolo

proactivo OLSR, dado que forma parte conjunta e inseparablemente de los protocolos de autoconfiguración especificados en los capítulos 6 y 7 de esta memoria.

# Capítulo 5

# Seguridad en Redes Móviles Ad Hoc

Este capítulo está dedicado a la problemática de la seguridad en las redes móviles ad hoc. Se comienza viendo las principales amenazas a la seguridad relativas a la autoconfiguración de direcciones en redes móviles ad hoc. Posteriormente, se introduce la necesidad de añadir extensiones seguras en la mayoría de los protocolos de encaminamiento para redes móviles ad hoc dado que la seguridad no fue una premisa de diseño de los mismos. Seguidamente se describen algunos ataques a los que está expuesto el protocolo de encaminamiento OLSR. A continuación se hace una revisión de los principales trabajos sobre seguridad en redes móviles ad hoc, enfocando la misma en la autoconfiguración segura de direcciones y en el encaminamiento seguro de OLSR, por estar íntimamente relacionado con los protocolos de autoconfiguración desarrollados en el presente trabajo. El capítulo finaliza con una breve síntesis de lo expuesto en el mismo.

## 5.1 Introducción

El diseño de protocolos de autoconfiguración y/o encaminamiento en redes móviles ad hoc no suele contemplar, en la mayoría de los casos, entornos hostiles. En otras palabras, en su especificación se asume la confianza en todos los nodos de la red, por lo que en el caso de comportamientos maliciosos la red se muestra claramente insegura, siendo vulnerable a todo tipo de ataques.

## 5.2 Autoconfiguración Segura

La autoconfiguración resulta conveniente en la implementación de una red móvil ad hoc. Sin embargo, puede sacrificar la seguridad de la red si no hay un mecanismo de autoconfiguración confiable. Por ejemplo, un atacante puede engañar a un nodo y secuestrar su tráfico.

Las redes móviles ad hoc constituyen una técnica novedosa que permite a una colección de nodos móviles comunicarse instantáneamente sin la existencia de ninguna infraestructura predefinida. Para asegurar que los paquetes puedan ser adecuadamente encaminados en una red móvil ad hoc, cada nodo debería configurarse con una dirección única. Debido a que la configuración manual no es siempre posible y presenta algunos inconvenientes, un proceso de configuración de direcciones dinámico y automático es deseable en la implementación de las redes móviles ad hoc.

Los protocolos propuestos en la literatura adolecen de un mecanismo de autenticación de direcciones. Consecuentemente, un nodo malicioso puede libremente engañar a cualquier nodo configurado y secuestrar su tráfico o evitar que un nuevo nodo pueda configurarse enviando falsos mensajes de conflicto de direcciones.

Seguidamente se describen las cuatro principales amenazas a la seguridad relativas a la autoconfiguración de direcciones en redes móviles ad hoc [WRN05] (otras amenazas como el ataque para consumir recursos y el encaminamiento seguro quedan fueras de esta clasificación):

- **Ataque de suplantación de identidad (*Address Spoofing Attack*)**: Sin un mecanismo de autenticación un nodo malicioso puede libremente elegir cualquier nodo configurado como víctima, suplantar su dirección IP y secuestrar su tráfico.
- **Ataque de Falso Conflicto de Dirección (*False Address Conflict Attack*)**: Un nodo malicioso puede transmitir intencionadamente un falso mensaje de conflicto de direcciones a una posible víctima. Ya que la víctima no puede verificar la autenticidad del conflicto de direcciones reportado puede tener que dejar su dirección actual y buscar una nueva.
- **Ataque de Agotamiento de Dirección (*Address Exhaustion Attack*)**: Un atacante puede maliciosamente reclamar tantas direcciones IP como existan. Si todas las direcciones IP válidas son consumidas por el atacante un nodo recién llegado no podrá obtener una dirección IP y, consecuentemente, evitará su entrada en la red.
- **Ataque de Respuesta Negativa (*Negative Replay Attack*)**: La asignación de una nueva dirección puede requerir la aprobación de todos los nodos configurados. Un atacante puede continuamente enviar respuestas negativas evitando que un nodo recién llegado obtenga una dirección.

## 5.3 Encaminamiento Seguro

Las redes móviles ad hoc están formadas por dispositivos móviles capaces de comunicarse entre sí sin necesidad de recurrir a una infraestructura de red preexistente. La mayoría de los protocolos de encaminamiento para este tipo de redes están diseñados sin tener en cuenta el posible comportamiento malicioso de alguno de los nodos, lo que puede ser aprovechado para vulnerar la seguridad de la red. Consecuentemente, es común añadir extensiones de seguridad a posteriori.

Dada la complejidad que supone resolver el problema de la seguridad de los diferentes protocolos de encaminamiento que se utilizan (sólo limitándose a los estándares de encaminamiento ya supondría una tarea inabordable) se centrará la atención en el protocolo de encaminamiento OLSR ya que juega un papel decisivo en el protocolo de autoconfiguración desarrollado en la memoria.

### 5.3.1 Ataques a OLSR

OLSR es un protocolo de encaminamiento que, en su especificación, asume la confianza de todos los nodos de la red, por lo que es vulnerable a diferentes clases de ataques.

Como su nombre indica, el protocolo OLSR emplea un mecanismo de inundación optimizado para difundir la información de los nodos y construir la topología de la red. Cada nodo selecciona un conjunto de nodos vecinos como MPR, que son los responsables de retransmitir el tráfico de control que se difunde en la red. Para realizar este proceso de

difusión se emplean los mensajes de control *Hello* y TC, especificados en las Figuras 5.1 y 5.2, respectivamente.

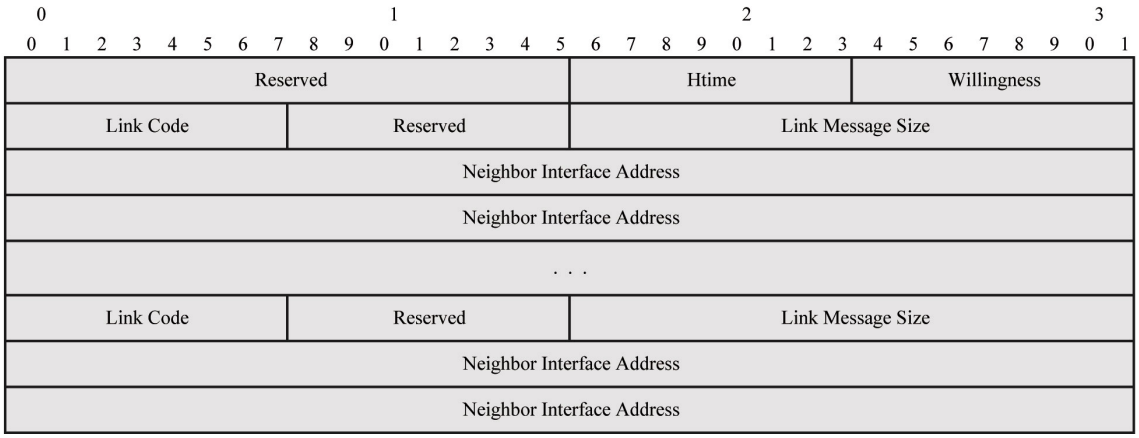



Figura 5.1: Formato del mensaje Hello del protocolo OLSR

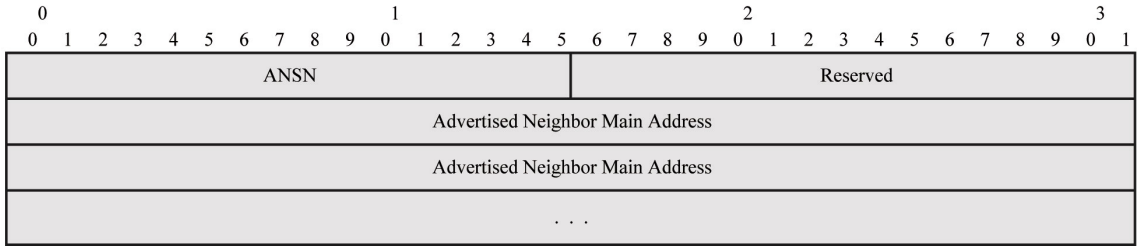



Figura 5.2: Formato del mensaje TC del protocolo OLSR

Los mensajes *Hello* permiten conocer a los vecinos de un nodo y calcular los nodos MPR. Los mensajes *Hello* sólo se intercambian periódicamente entre nodos vecinos, proporcionando las listas de vecinos de cada nodo. Una de las listas contiene los vecinos que han sido *escuchados* por el nodo actual, pero que todavía no tienen confirmado el enlace bidireccional. Otra lista está formada por los vecinos que han establecido una comunicación bidireccional (enlace simétrico). La última lista está constituida por los nodos vecinos que han sido seleccionados por el origen del mensaje *Hello* para actuar como MPR.

Los mensajes TC se envían periódicamente a través de los MPR. Su finalidad es difundir la información de la topología a toda la red. Un mensaje TC contiene el conjunto de enlaces bidireccionales entre un nodo y sus vecinos. La lista de vecinos que incluye es el conjunto selector MPR, es decir, los nodos que han elegido al que emite el mensaje TC como MPR.

Existen otros mensajes de control con una funcionalidad diferente que no se tendrán en consideración, ya que la amplia mayoría de los ataques se centran en los anteriores. En efecto, los principales ataques [KNK+07] [AKG08] a los que OLSR está expuesto, están estrechamente relacionados con los anteriores mensajes de control, distinguiéndose los siguientes tipos:

- **Ataque de Generación de Mensaje Incorrecto o *Incorrect Message Generation* (IMG)**: Este ataque se produce cuando un nodo tiene un mal comportamiento en la generación de los mensajes, circunstancia ésta que puede ocurrir:

  a. Cuando hay suplantación de identidad (*Identity Spoofing*), es decir, un nodo se hace pasar por otro.

  b. Cuando un nodo anuncia información incorrecta de enlaces en los mensajes de control (*Link Spoofing*).

- **Ataque de Generación de Tráfico Incorrecto o *Incorrect Traffic Generation* (ITG)**. Este ataque genera tráfico incorrecto aunque los mensajes sean correctos. Se distinguen, a su vez, dos tipos:

  a. Ataques de Repetición (*Replay Attacks*) en los que un nodo intruso reenvía mensajes obsoletos generados en la red.

  b. Ataques de Agujero de Gusano (*Wormhole Attacks*) [HPJ06], difíciles de detectar, en lo que participan dos o más nodos que colaboran para crear un túnel entre ellos para tener acceso directo a la red. Una vez creado el túnel (el agujero del gusano), los atacantes encapsulan los mensajes recibidos y realizan el intercambio de estos mensajes a través del túnel, impidiendo que los nodos intermedios reciban los mensajes de control.

## 5.4 Trabajos Relacionados

En este apartado se hace una recopilación del estado de arte de los trabajos más representativos relativos a la seguridad en redes móviles ad hoc. Con objeto de facilitar una mejor comprensión se presentarán agrupados en torno a la problemática en la que se centren.

### 5.4.1 Seguridad en Autoconfiguración

Lamentablemente, la literatura relativa a la *seguridad en el proceso de autoconfiguración* es muy escasa y la existente o bien propone soluciones (que no implementa en muchos casos) muy complejas y, consecuentemente, no son adecuadas y, desde luego, no escalables, o bien abordan aspectos muy específicos, no siendo válidas globalmente.

No obstante lo anterior, cabe destacar algunos trabajos, tales como:

[PPM+03] propone una arquitectura modular y distribuida para un sistema de detección de intrusiones o *Intrusion Detection System* (IDS) en entornos de redes móviles ad hoc. La principal característica de esta propuesta consiste en el uso, en cada nodo de la red móvil ad hoc, de un IDS Local o *Local IDS* (LIDS). Estos LIDS cooperan entre sí mediante el uso de agentes móviles. El sistema se comporta bastante bien para detectar ataques conocidos, esto es, como sistema de detección de intrusos basado en firmas, detectando muchos de ellos incluso en tiempo real. Sin embargo, deben mejorar todo lo relativo a los ataques anómalos (o desconocidos) que, por desgracia, son cada vez más numerosos.

[ACL+03] propone un esquema para autenticar mensajes OLSR en una aproximación extremo a extremo, de forma que los nodos que reciben mensajes OLSR pueden autenticar el nodo que generó el mensaje original. Sin embargo, los ataques de repetición son todavía posibles. Para evitar este tipo de ataques desarrolla otra aproximación basada en marcas o sellos de tiempo distribuidos para verificar si un mensaje es obsoleto o no.

[PH03] provee seguridad en el encaminamiento de estado de enlace mediante el uso de primitivas asimétricas. Para ello asume que cada nodo de la red tiene un par de claves pública/privada y que difunde en modo *broadcast* su certificado de clave pública al resto de los nodos que están dentro de N saltos. Estos mensajes enviados en modo *broadcast* pueden ser periódicos o no, dependiendo de los cambios de la topología de la red, para posibilitar que un nuevo nodo pueda conocer la clave al entrar en la zona. Pese a utilizar certificación distribuida la sobrecarga introducida también es importante.

[CO04] propone una adaptación del protocolo de autoconfiguración de Mohsin y Prakash [MP02] para que los nodos puedan trabajar de forma segura. La propuesta reduce la sobrecarga del protocolo de Mohsin y Prakash para permitir el uso de funciones de autenticación. A pesar de que el protocolo funciona relativamente bien, no logra garantizar la seguridad durante la fusión de redes. Sin embargo, el esquema propuesto es interesante y puede utilizarse como complemento a otros esquemas de seguridad (tanto en autoconfiguración como en encaminamiento).

[HTR+04] presenta una aproximación salto a salto, donde cada nodo firma los paquetes OLSR que son transmitidos, descartando la firma del nodo anterior. Esta firma sólo verifica que el nodo que reenvía el mensaje es el mismo que lo firma, pero no verifica la autenticidad del mensaje original. Las soluciones implementadas usan claves compartidas (simétricas) para la creación de la firma y la verificación. Sin embargo, no mencionan cómo se realiza la administración/intercambio de claves o la autenticación inicial. Asimismo, proponen un método para evitar los ataques de repetición utilizando marcas o sellos de tiempo, si bien la sobrecarga aumenta de forma importante.

[RACM04a] diseña otra extensión extremo a extremo en la que los nodos envían mensajes OLSR junto a un mensaje *Advanced Signature* (ADVSIG). Los nodos que anuncian enlaces firman los mensajes con el fin de poder autenticar el nodo que originó el mensaje. Se establecen mecanismos para mejorar la seguridad del protocolo contra ataques externos así como procedentes de nodos comprometidos, si bien de forma muy costosa en términos de sobrecarga.

[RACM04b] proporciona un método que incluye la posición geográfica del nodo emisor en los mensajes de control al mismo tiempo que evalúa la similitud de los enlaces. Esta técnica tiene el inconveniente de que requiere hardware específico como dispositivos *Global Positioning System* (GPS) y antenas direccionales.

[HHW04] presenta un modelo para establecer relaciones de confianza entre los nodos de una red móvil ad hoc.

[HM05] propone un modelo de confianza que mejora la seguridad de los esquemas de inicialización.

Si bien estos dos últimos trabajos son de interés, son también incompletos no precisando muchos detalles en los mismos.

Mención especial merece [WRN05] que presenta un esquema de autoconfiguración con auto-autenticación. El esquema propuesto une la dirección de un nodo con una clave pública mediante una función hash unidireccional. Por tanto, el propietario de una dirección puede usar la correspondiente clave pública para unilateralmente auto-autenticarse a sí mismo. Tal autenticación unilateral contrarresta en gran medida los ataques comentados anteriormente.

Seguidamente se describe con mayor detalle:

Se asume que la red móvil ad hoc es una red completamente privada. Por tanto, todo el espacio de direcciones está disponible para autoconfiguración. En otras palabras, los 32 bits (en IPv4) o los 128 (en IPv6) pueden utilizarse para direccionar los nodos en la red móvil ad hoc. Se asume que los nodos en la red se sincronizan periódicamente.

En este esquema un nodo no configurado denominado A se quiere unir a una red móvil ad hoc existente o iniciar una nueva. En primer lugar genera aleatoriamente un par de claves pública y privada y una clave secreta, e inmediatamente almacena estas claves en un lugar seguro. A continuación, el nodo A calcula un hash de 32 bits (en IPv4) o de 128 (en IPv6) de su clave pública. Posteriormente, el nodo A utiliza temporalmente el valor resultante como su dirección IP, inicia un temporizador y difunde un mensaje *Duplicate Address Probe* (DAP).

Si un nodo configurado denominado B encuentra que la dirección IP en un mensaje DAP recibido es igual que la suya verificará la autenticidad de este mensaje DAP. El nodo B comprueba primero si las direcciones IP coinciden. Si es así, el nodo B verifica la firma usando la clave pública recibida. Si la firma se verifica correctamente, el nodo B verifica si su clave pública coincide con la de A y si el sello de tiempo incrustado en el mensaje DAP también coincide (para prevenir la posibilidad de un ataque de repetición o *Relay Attacks*). Si al menos una de las dos comparaciones no se verifica, el nodo B difunde en *broadcast* un mensaje *Address Conflict Note* (ACN) para comunicar al nodo A el correspondiente conflicto de dirección. En otro caso el nodo B descarta el mensaje DAP recibido.

Si ningún mensaje ACN se recibe antes de que el temporizador expire, el nodo A asume que la dirección IP correspondiente no está en uso y empieza a utilizarla.

Si el nodo A recibe un mensaje ACN de otro nodo, por ejemplo del nodo B, verifica primero este mensaje ACN antes de intentar otra dirección IP. El nodo A examina el sello de tiempo para ver si el mensaje ACN recibido está caducado. Si no lo está, verifica o comprueba si las direcciones IP coinciden y si la firma es correcta. Si todas las verificaciones son positivas el nodo A puede estar seguro de que la dirección IP correspondiente está en uso por otro nodo. Entonces el nodo A tiene que repetir el procedimiento de autoconfiguración con otro par de claves pública y privada. En otro caso, el nodo A descarta el mensaje ACN recibido.

Es posible que dos nodos configurados puedan tener la misma dirección IP después de una unión de particiones. En este caso, cuando el conflicto de direcciones se detecta por alguno de los mecanismos existentes, uno de los dos nodos, por ejemplo el nodo A, difunde en *broadcast* un mensaje ACN. Una vez que recibe el mensaje ACN el otro nodo, en este caso el B, comprueba primero la autenticidad de este ACN tal y como se describió antes. Si el resultado de la verificación es positivo el nodo B comprueba entonces si es la fuente de este mensaje ACN para evitar un ataque de repetición. El nodo B comprueba si la clave pública contenida en el mensaje es la misma que la suya y comprueba asimismo el sello de tiempo. Si todos los resultados son positivos, el nodo B trata este mensaje ACN como una repetición y lo descarta. En otro caso, el nodo B deja su dirección IP actual y comienza otro proceso de autoconfiguración de direcciones.

Dos nodos con conflicto de direcciones pueden recibir mutuamente un mensaje ACN. Para evitar que simultáneamente cambien sus direcciones IP, sólo el nodo que tenga el menor sello de tiempo en el mensaje DAP debería cambiar su dirección IP.

### 5.4.2 Seguridad en OLSR

En cuanto a los *trabajos más representativos relativos a la seguridad en OLSR* cabe señalar que existen diversas técnicas para proporcionar integridad al protocolo. Una de las más extendidas es el uso de firmas digitales para la autenticación de los mensajes de encaminamiento OLSR, diferenciándose dos variantes o aproximaciones: salto a salto (*hop-by-hop*) y extremo a extremo (*end-to-end*).

[NA08] expone un esquema para la detección y prevención de los ataques de agujero de gusano. Esta técnica se basa primero en detectar los posibles enlaces en los que se produce

el ataque para distinguir a continuación entre los enlaces en los que se produce un ataque de agujero de gusano y los que son correctos. Para ello se intercambian paquetes cifrados entre los dos supuestos vecinos que realizan el ataque. Esta solución es independiente de cualquier sincronización de tiempo o información de localización, pero tiene el inconveniente de aumentar la sobrecarga debido al intercambio de mensajes para la detección del ataque.

[TT09] presenta un protocolo seguro de autoconfiguración para redes móviles ad hoc. La propuesta utiliza funciones criptográficas (funciones hash y criptografía simétrica) para hacer frente a todos los posibles ataques a la autoconfiguración. Sin embargo, el protocolo presupone que cada nodo tiene pre-distribuida una clave secreta y no especifica nada sobre la distribución y gestión de claves.

Finalmente, [ZMN09] propone un protocolo seguro de autoconfiguración mediante el uso de infraestructuras de clave pública o *Public Key Infrastructure* (PKI). El protocolo resuelve cómo distribuir de forma segura la clave pública de un nuevo nodo a todos (o a la mayoría de) los miembros de la red móvil ad hoc. Sin embargo no especifica cómo los miembros ya existentes pueden distribuir sus claves públicas a los nuevos nodos de forma segura.

## 5.5 Síntesis del Capítulo

El principal objetivo de este capítulo ha sido abordar la problemática de la seguridad en las redes móviles ad hoc, aspecto éste bastante deficiente en la literatura especializada. Se ha visto como el diseño de protocolos de autoconfiguración y/o encaminamiento en redes móviles ad hoc no suele contemplar, en la mayoría de los casos, entornos hostiles, lo que hace necesario la definición de extensiones de seguridad a posteriori de los protocolos más importantes. También se han señalado algunos de los ataques más comunes a los que está expuesto el protocolo OLSR, dada la importancia de este protocolo en los protocolos de autoconfiguración desarrollados en el presente trabajo (capítulos 6 y 7) como en la extensión de seguridad de este protocolo definida en el capítulo 8. Asimismo, se ha hecho una revisión de los trabajos más importantes sobre seguridad en autoconfiguración de redes móviles ad hoc así como de las principales extensiones de seguridad de OLSR.

# Capítulo 6

# Protocolo D2HCP

Este capítulo presenta el diseño y especificación de un protocolo de autoconfiguración para redes móviles ad hoc. El protocolo se ha denominado *Protocolo de Configuración de Host Dinámico y Distribuido* o, más comúnmente, D2HCP, como se cita en lo sucesivo, expresión que corresponde al acrónimo de su denominación inglesa *Distributed Dynamic Host Configuration Protocol*. El capítulo comienza describiendo las principales características del protocolo D2HCP y analizando las principales diferencias respecto al protocolo de Mohsin y Prakash, protocolo en el que se inspira el diseño de D2HCP. A continuación se detallan las estructuras de datos usadas por el protocolo. Seguidamente se analizan la entrada y salida de nodos, el proceso de sincronización del protocolo y la inicialización de la red. Posteriormente, se proporciona una completa especificación del protocolo, detallando los mensajes, los temporizadores y el diagrama de estado de los nodos cliente y servidor en D2HCP. El capítulo finaliza con una breve síntesis de lo expuesto en el mismo.

## 6.1 Generalidades

D2HCP es un protocolo de autoconfiguración que gestiona la entrada y salida de nodos en redes móviles ad hoc.

El protocolo hace que los nodos de una red móvil ad hoc colaboren entre sí para gestionar la asignación de direcciones IP únicas y correctas de forma distribuida. Todos los nodos de la red tienen el mismo papel, no existiendo un tipo de nodo especial que centralice la gestión de la misma.

Los nodos cuentan con un sistema de sincronización que se apoya en el protocolo de encaminamiento OLSR. Gracias a este mecanismo la sincronización se lleva a cabo de forma pasiva, monitorizando el protocolo de encaminamiento mencionado, por lo que se genera una sobrecarga nula en el tráfico de la red con respecto al generado por el protocolo OLSR.

Al ser todos los nodos responsables de gestionar la entrada de cualquier otro nuevo nodo a la red, esta operación puede realizarse rápidamente. Un nodo que desea entrar a la red intenta contactar con cualquier nodo ya perteneciente a ella, pudiendo recibir respuesta de varios nodos. Esto hace que las posibilidades de entrar a formar parte de la red con éxito sean altas, debido a la alta disponibilidad y a la redundancia que supone la gestión distribuida.

## 6.2 Consideraciones de Diseño

Como se ha comentado en el apartado 3.5.1, en el protocolo de Mohsin y Prakash [MP02] se tiene que:

- Por cada nodo de la red hay que conocer la siguiente información: su dirección IP, los bloques de direcciones libres *Free_IP_blocks*, el nodo *father* que le suministró la dirección IP y su marca de tiempo TS.
- Conviene recordar que *Free_IP_blocks* se refiere a varios bloques disjuntos de direcciones IP. Los campos *father* y TS se usan para determinar quién es el responsable de recuperar las direcciones libres que un nodo deja disponibles al abandonar la red. Este responsable es el nodo *father*, o en caso de no estar disponible, el nodo de la red con menor TS. El hecho de que cada nodo pueda tener muchos bloques diferentes de direcciones libres hace que no se pueda determinar cuánto puede llegar a ocupar el campo *Free_IP_blocks*.
- Se concluyó que *si se cambiase el modo en el que se recuperan las direcciones de un nodo que sale de la red, los campos father y TS podrían omitirse.*

Las diferencias de D2HCP con respecto a su predecesor se resumen en los siguientes puntos:

- Cada nodo tiene como información asociada para la autoconfiguración un único bloque de direcciones IP libres contiguas, que incluye la propia dirección IP del nodo.
- El responsable de recuperar las direcciones IP que un nodo que abandona la red deja disponibles es aquel que puede unir por la derecha ese bloque libre con el suyo.

  Esto no será posible cuando el bloque que se debe recoger contiene la dirección más baja de la red. En ese caso, el nodo que recoge el bloque es aquel que puede añadirlo al suyo por la izquierda.
- Al dividir las direcciones libres en dos bloques para entregar uno de ellos a un nuevo nodo que entra a la red, el nodo que hace de servidor entrega al cliente el sub-bloque que no contiene su propia dirección IP.

Con este nuevo esquema se consiguen varias mejoras:

- No son necesarios los campos *father* y TS, ya que el propio bloque de direcciones libres de cada nodo determina quién es el responsable de recogerlo en caso de que un nodo abandone la red.
- Se garantiza además que cada nodo tiene un solo bloque de direcciones IP libres en todo momento, siendo más manejable su gestión pues se conoce cuánto ocupará esta información y se garantiza que este tamaño será constante. Esto es importante para poder estudiar los requisitos de memoria de los nodos o la sobrecarga que supondría su transmisión por la red.
- En el protocolo original cada nodo debe comprobar periódicamente mediante mensajes *ping* si siguen activos tanto el nodo padre como los nodos hijos. Se entiende por nodo padre el que le suministró su dirección IP y por nodos hijos los nodos a los que se les ha facilitado una dirección IP y un bloque de direcciones libres.

Con las mejoras propuestas se consigue que los nodos no tengan que comprobar activamente si otros nodos siguen participando en la red. Será en el momento de detectar que otro nodo ha abandonado la red, cuando cada nodo comprobará si el bloque de direcciones que queda disponible debe ser recogido por él mismo o por otro de los nodos de la red.

## 6.3 Estructuras de Datos

Las estructuras de datos de este protocolo pueden clasificarse en las propias que maneja el mecanismo de autoconfiguración y las pertenecientes al protocolo de encaminamiento OLSR.

OLSR almacena internamente una *tabla de encaminamiento* o *tabla de rutas* que es actualizada periódicamente. Esta tabla contiene información sobre la ruta a cada nodo, almacenada en los siguientes campos:

- **R_dest_addr**: Dirección IP del nodo destino.
- **R_next_addr**: Dirección IP del siguiente salto en la ruta.
- **R_dist**: Distancia al nodo destino.
- **R_iface_addr**: Dirección IP de la interfaz de salida al nodo destino.

Las estructuras de datos necesarias para el mecanismo de autoconfiguración son:

- Direcciones IP de las interfaces del nodo.
- Máscara de red.
- Tabla de bloques libres de cada nodo de la red (*Free_IP_Blocks*). En la Tabla 6.1 se muestra un ejemplo de esta tabla.

Tabla 6.1: Ejemplo de tabla Free_IP_Blocks

| IP | Bloque de Direcciones |
|---|---|
| .1 | .1 - .64 |
| .128 | .128 - .254 |
| .65 | .65 - .127 |

## 6.4 Entrada y Salida de Nodos

El protocolo usa un número concreto de mensajes para cada operación. Todas las operaciones están definidas buscando un óptimo funcionamiento y una latencia mínima.

### 6.4.1 Entrada de Nodos

La entrada de un nodo a la red implica la necesidad de encontrar un nodo que actúe como servidor. Una vez encontrado uno, éste le facilita la entrada proporcionándole un bloque de direcciones IP y una tabla *Free_IP_Blocks* que representa el estado de los nodos de toda la red. Hasta que el nodo no tenga asignada una dirección IP su comunicación con los nodos servidores será a través de la capa MAC.

Este mecanismo de configuración utiliza 4 tipos de mensajes en la mayoría de los casos. Si no hay nodos en su rango con direcciones IP libres se usan 6 tipos de mensajes en total.

En la Figura 6.1 se muestra el esquema de intercambio de mensajes que se explica a continuación:

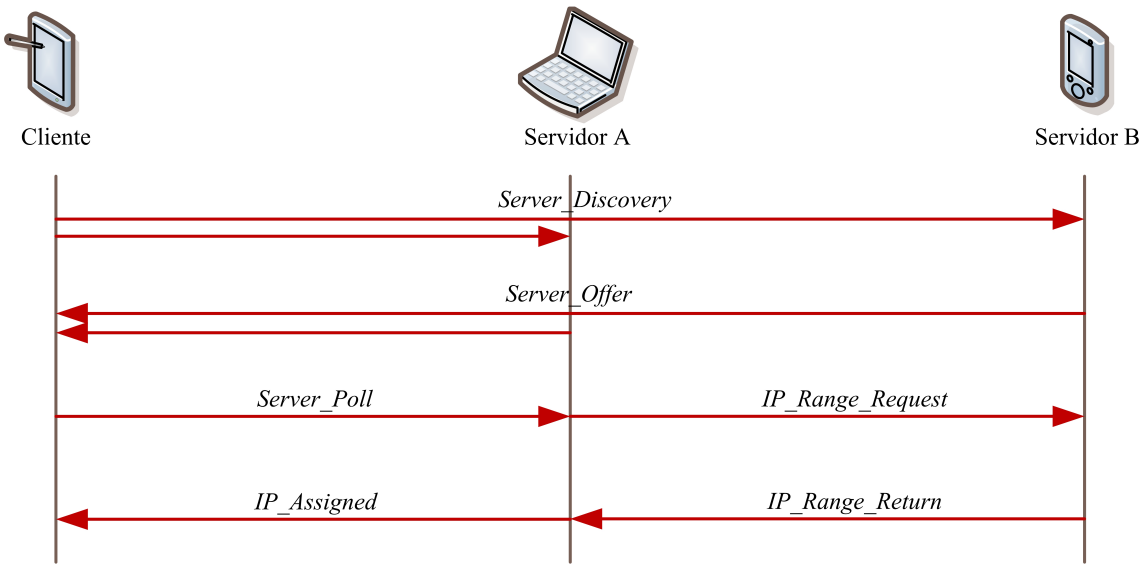



Figura 6.1: Intercambio de mensajes en el proceso de entrada de un nodo a la red

#### Server_Discovery

El nodo cliente que desea entrar en la red inicia el proceso con un mensaje de este tipo. Se transmite por la capa MAC con la dirección de difusión como destino. En el mensaje se indica el número de direcciones IP que se necesitan (que será igual al número de interfaces).

Si el nodo dispone de más de una interfaz de red, el mensaje se transmite por todas ellas usando el campo ID para que las diferentes interfaces no se confundan con varios nodos.

#### Server_Offer

Los nodos de la red que reciben el mensaje *Server_Discovery* responden con este mensaje, también usando la capa MAC, en el que ofrecen un número de direcciones IP. El número de direcciones ofrecidas es la mitad del rango disponible.

El mensaje *Server_Discovery* incluye un campo *Count* que indica cuántos intentos se han realizado por parte del cliente. Dependiendo de su valor los nodos servidores se comportan del siguiente modo:

- *Count* = 1: El nodo servidor responde con un mensaje *Server_Offer* si dispone de direcciones suficientes con los campos R = 1 (*Ready*) y L = 1 (*Local*), indicando que puede asignar inmediatamente las direcciones IP ofrecidas y que éstas pertenecen al propio nodo.

- *Count* = 2: El nodo servidor responde con un mensaje *Server_Offer* con los campos R = 0 y L = 1 cuando tiene suficientes direcciones IP pero no puede asignarlas inmediatamente.

- *Count* > 2: El nodo servidor responde con un *Server_Offer* con los campos R = 1 y L = 0 cuando no dispone de suficientes direcciones IP libres pero existe disponibilidad inmediata de direcciones IP en un tercer nodo.

**Server_Poll**

Tras un tiempo de escucha el nodo cliente recibe varios mensajes *Server_Offer*. De no ser así, vuelve a intentarlo.

Ordena los mensajes recibidos según los siguientes criterios:

- Se descartan los servidores que no estén disponibles, es decir, con $R = 0$. Los mensajes *Server_Offer* con R = 0 no se usan para poder responder con un *Server_Poll*, pero tienen la función de informar al cliente de su existencia aunque en ese momento no puedan facilitarle el acceso.

- Se da prioridad a las direcciones locales, esto es, tienen preferencia los mensajes con el campo L = 1.

- Se ordenan por número de direcciones ofrecidas de mayor a menor.

Según ese orden de preferencia, envía al primer servidor un mensaje *Server_Poll* (de nuevo, por la capa MAC) para indicarle que le ha elegido para que le asigne un bloque de direcciones IP libres.

**IP_Range_Request**

Si las direcciones ofrecidas por el nodo servidor no eran propias, sino de un tercer nodo de la red, con este mensaje se le solicitan formalmente. Al ser una comunicación entre dos nodos ya configurados correctamente se realiza en la capa IP.

**IP_Range_Return**

Ese tercer nodo de la red autoriza al nodo que envió el mensaje *IP_Range_Request* a asignar al nodo cliente el bloque de direcciones indicado en este mensaje. También es un mensaje enviado por IP.

**IP_Assigned**

Tras recibir el *Server_Poll*, si las direcciones ofrecidas eran del propio nodo servidor o tras el mensaje *IP_Range_Return*, en caso de que se haya tenido que pedir las direcciones a un tercer nodo, el nodo servidor envía este mensaje al cliente. Este mensaje es transmitido por la capa MAC.

En él se indica el bloque de direcciones libres que se le asigna al cliente y la tabla *Free_IP_Blocks* que representa el estado de la red. La tabla que se transmite en este mensaje no refleja la entrada del nodo cliente.

Tras este intercambio de mensajes el nodo cliente elige como su dirección IP la primera del bloque que se le ha asignado. En caso de tener más de una interfaz de red usará las primeras del bloque en orden y será la primera de todas la que use como dirección principal para identificar al nodo.

### 6.4.2 Salida de Nodos

El mecanismo de salida de nodos no requiere del intercambio de ningún mensaje. El nodo que quiera abandonar la red no tiene que avisar a ningún otro nodo de su salida, evitando la sobrecarga que estos mensajes producen. El resto de nodos de la red se darán cuenta de la salida del nodo mediante las actualizaciones periódicas de rutas que el protocolo OLSR realiza cada cierto tiempo. Observarán que se ha perdido la ruta hacia ese nodo eliminándolo de su tabla *Free_IP_Blocks* y añadiendo su bloque de direcciones libres al nodo que corresponda según se ha explicado en el apartado 6.2.

## 6.5 Sincronización

La sincronización se lleva a cabo monitorizando la *tabla de encaminamiento* del protocolo de encaminamiento OLSR. La entrada o salida de un nodo se detecta cuando OLSR añade una nueva ruta a su tabla de encaminamiento o borra una de las existentes. Al detectar la entrada o la salida de un nodo, se actualiza la tabla *Free_IP_Blocks* localmente sin intercambiar ningún mensaje. Para ello se sigue la siguiente operativa:

- El responsable de recuperar las direcciones IP que un nodo que abandona la red deja disponibles es aquel que puede unir por la derecha ese bloque libre con el suyo.

  Esto no será posible cuando el bloque que se debe recoger contiene la dirección más baja de la red. En ese caso, el nodo que recoge el bloque es aquel que puede añadirlo al suyo por la izquierda.

- Al dividir las direcciones libres en dos bloques para entregar uno de ellos a un nuevo nodo que entra a la red, el nodo que hace de servidor entrega al cliente el sub-bloque que no contiene su propia dirección IP.

Cuando se detecta la salida de un nodo se debe eliminar su entrada y actualizar la del nodo a quien corresponden las direcciones IP que han quedado disponibles.

Al detectar el ingreso de un nuevo nodo se crea una nueva entrada en la tabla para él y se actualiza el bloque de direcciones libres del nodo que le facilitó su dirección IP. Para saber quién fue ese nodo que actuó como servidor basta con buscar qué nodo tiene la dirección IP del nuevo nodo en su bloque de direcciones libres.

### 6.5.1 Elección de OLSR como Mecanismo de Sincronización

Después de un análisis de los protocolos de encaminamiento más conocidos se decidió usar OLSR como base para la sincronización del protocolo de autoconfiguración por varios motivos:

- Al ser un protocolo proactivo con refresco de las rutas de forma periódica permite conocer la topología de la red en todo momento. Aunque algún paquete de actualización se pierda, las rutas se actualizarán igualmente con las siguientes transmisiones.
- OLSR aporta la infraestructura para distribuir información a toda la red usando MPR, mecanismo que podría ser utilizado para hacer llegar nuevos mensajes de control a todos los nodos de la red.
- El estar optimizado para funcionar en redes a gran escala también influyó en la decisión de utilizarlo, ya que se pretendía someter al protocolo de autoconfiguración resultante a pruebas exhaustivas en complejos escenarios.

Para terminar con el análisis de OLSR se intentó concretar qué información se podía deducir de este protocolo de encaminamiento, monitorizando su comportamiento o sus estructuras de datos, sin modificar ni su comportamiento ni sus paquetes.

Se centró el interés en interpretar los datos que OLSR maneja y los cambios que suceden con estos datos.

Debido a que se tiene información actualizada sobre la topología de la red puede determinarse la entrada o la salida de cualquier nodo de forma inmediata y sencilla. Además, se tiene información acerca de esos nodos como, por ejemplo, su dirección IP.

Aunque parezca información escasa, más adelante se verá que será más que suficiente para desarrollar ideas más complejas.

También se vio que monitorizando las estructuras de datos que OLSR maneja se puede conocer el conjunto de nodos considerados vecinos, es decir, con los que se tiene un enlace directo o que se encuentran a un salto de distancia.

De nuevo, puede parecer información poco relevante, pero permite distinguir entre nodos vecinos o lejanos para controlar el comportamiento de mecanismos más complejos. Por ejemplo, esta información puede ser usada para decidir enviar ciertas peticiones sólo a los nodos vecinos con el fin de tener respuestas inmediatas y evitar inundar toda la red con mensajes de control.

La solución que se propone a la necesidad de un mecanismo de sincronización para el protocolo D2HCP es integrar en parte el protocolo de encaminamiento OLSR.

Al integrar la autoconfiguración con OLSR se consigue aprovechar el mecanismo provisto por este último para detectar los cambios en la red. Recuérdese que OLSR es un protocolo de encaminamiento proactivo que mantiene información actualizada sobre las rutas hacia todos los nodos de la red. Por tanto, es capaz de detectar la entrada de un nuevo nodo a la red (descubrimiento de una ruta nueva) y la salida de nodos (eliminación de una de las rutas conocidas).

Por cada nodo de la red OLSR mantiene una entrada en su *tabla de encaminamiento*. Esta tabla consta de los campos:

***R_dest_addr*** ***R_next_addr*** ***R_dist*** ***R_iface_addr***

La Tabla 6.2 muestra un ejemplo donde se han omitido los campos referentes a las rutas.

En el protocolo propuesto se necesita conocer el bloque de direcciones IP que tiene asignado cada nodo de la red, de modo que cada nodo tiene almacenada una tabla similar a la Tabla 6.3.

Puede verse la relación entre una y otra de forma clara. En cada tabla se tiene una entrada por cada nodo de la red. También se cumple que ambas tablas deben estar actualizadas constantemente: deben reflejar la entrada o salida de cualquier nodo de la red.

Tabla 6.2: Ejemplo de tabla de encaminamiento de OLSR

| **R_dest_addr** | **R_next_addr** | **R_dist** | **R_iface_addr** |
|---|---|---|---|
| .1 | ... | ... | ... |
| .128 | ... | ... | ... |
| .65 | ... | ... | ... |

Tabla 6.3: Tabla Free_IP_Blocks

| **IP** | **Bloque de Direcciones** |
|---|---|
| .1 | .1 - .64 |
| .128 | .128 - .254 |
| .65 | .65 - .127 |

Por tanto, ya que OLSR se encarga de conocer la topología de la red en todo momento, basta con monitorizar su tabla de encaminamiento para detectar cuándo un nodo se une o abandona la red. Es decir, la actualización de la tabla *Free_IP_Blocks* sigue a la actualización de la tabla de encaminamiento realizada esta última por OLSR.

A continuación se muestra un ejemplo que ilustra el funcionamiento y la actualización de los campos. Se comentan las estructuras relacionadas con la sincronización, omitiendo los mensajes necesarios para realizar estos pasos.

La Tabla 6.4 muestra todos los nodos de una red.

Tabla 6.4: Tabla Free_IP_Blocks con un nodo

| **IP** | **Bloque de Direcciones** |
|---|---|
| .1 | .1 - .254 |

Existe un único nodo con 254 direcciones IP libres.

Esta es la tabla que maneja el protocolo internamente.

Como puede observarse, la propia dirección del nodo está dentro del bloque de direcciones IP libres. Más adelante se verá por qué esto es necesario y cómo se garantiza que no sea un problema.

Supóngase ahora que un nuevo nodo entra en la red y que obtiene una dirección IP y un bloque de direcciones IP libres del nodo con la dirección IP .1.

El bloque de direcciones libres [.1 - .127] se divide en dos sub-bloques de 127 direcciones cada uno: [.1 - .127] y [.128 - .254].

El nodo con la dirección .1 debe entregar uno de estos dos sub-bloques al nuevo nodo cliente, entregando el que no contiene su propia dirección (.1). Por tanto, el nodo que entra en la red tiene asignado el rango de direcciones libres [.128 - .254], del que toma la primera dirección para su propio uso.

En el caso de que el nuevo nodo tenga más de una interfaz de red participando en la misma red móvil ad hoc, usará tantas direcciones como interfaces. Esas direcciones serán las primeras del rango. La primera de las direcciones de sus interfaces será además su dirección principal, que identificará al nodo. Cada dirección secundaria está ligada a la dirección principal del nodo.

En este momento, el nuevo nodo .128 está correctamente configurado y comienza a utilizar el encaminamiento OLSR. Asimismo, el nodo .1 descubrirá gracias a OLSR una nueva ruta hacia el nodo .128.

Esta nueva ruta de la tabla de encaminamiento se añade a la tabla de autoconfiguración como se muestra en la Tabla 6.5.

Tabla 6.5: Tabla Free_IP_Blocks al ingresar el nodo .128 a la red

| **IP** | **Bloque de Direcciones** |
|---|---|
| .1 | .1 - .127 |
| .128 | .128 - .254 |

En este caso, .1 dispone de 127 direcciones libres que debe dividir en dos sub-bloques. Como no puede crear dos sub-bloques de 63.5 direcciones cada uno, creará los sub-bloques [.1 - .64] y [.65 - .127] de 64 y 63 direcciones, respectivamente. De nuevo, la dirección que está usando (.1) está contenida en el sub-bloque izquierdo, por lo que asigna el derecho al nuevo nodo (véase Tabla 6.6).

Tabla 6.6: Tabla Free_IP_Blocks al ingresar el nodo .65 a la red

| **IP** | **Bloque de Direcciones** |
|---|---|
| .1 | .1 - .64 |
| .128 | .128 - .254 |
| .65 | .65 - .127 |

Supóngase ahora que el nodo .128 abandona la red. Al dejar de participar en la red, las direcciones libres [.128 - .254] no serán asignadas por .128 a nuevos nodos.

Estas direcciones no pueden desperdiciarse, por lo que alguien tiene que recogerlas. El encargado de ello es el único nodo que puede añadir esas direcciones por la derecha al bloque de direcciones IP libres que ya tiene. Ese nodo es .65 (véase Tabla 6.7).

Conviene recordar que esta tabla es una estructura interna que maneja el protocolo. Cada nodo de la red tiene una copia y realiza las actualizaciones de forma local sin intercambiar ningún mensaje.

Tabla 6.7: Tabla Free_IP_Blocks al salir el nodo .128 de la red

| IP | Bloque de Direcciones |
|---|---|
| .1 | .1 - .64 |
| .65 | .65 - .254 |

Hay que destacar que el nodo .1 fue quien facilitó la entrada a la red al nodo .128. En el protocolo de Mohsin y Prakash el nodo .1 sería el padre de .128. Sin embargo, se ha visto que el encargado de recoger las direcciones que .128 deja disponibles es otro nodo diferente, determinado únicamente por el bloque de direcciones que tiene asignado.

Como puede observarse en la Tabla 6.7, el bloque de direcciones libres del nodo .65 ha crecido, pero sigue manteniéndose como un único bloque de direcciones contiguas.

En el caso de que .1 dejase la red, su bloque de direcciones [.1 - .64] debe ser recogido por otro nodo. En este caso, no hay ningún nodo que pueda añadir este bloque al suyo por la derecha, ya que la dirección más baja de la red (.1) está contenida en él.

Por tanto, sus direcciones tienen que ser recogidas por el nodo que pueda añadir este bloque que queda disponible al suyo por la izquierda. De nuevo es el nodo .65 el responsable (véase Tabla 6.8).

Tabla 6.8: Tabla Free_IP_Blocks al salir el nodo .1 de la red

| IP | Bloque de Direcciones |
|---|---|
| .65 | .1 - .254 |

Se observa una situación nueva: la dirección IP del nodo (.65) está contenida en la mitad de su bloque de direcciones IP libres, en lugar de ser la primera de él.

Si no se permitiese esto, el bloque no sería uno solo y se tendrían las direcciones libres [.1 - .64] + [.66 - .254] y ya no se cumpliría que cada nodo tiene un único bloque de direcciones libres.

En realidad, esto no supone un problema a la hora de realizar la división binaria de direcciones libres. La entrada de un nodo en este momento supondría que el nodo .65 ha de dividir su bloque de direcciones. Su propia dirección IP queda en el sub-bloque izquierdo y puede desprenderse de las direcciones [.128 - .254] sin problema.

En este punto la tabla quedaría como se muestra en la Tabla 6.9.

Si otro nodo entra en la red y es .65 de nuevo quien le proporciona su dirección IP, las direcciones IP libres tendrían que dividirse en los sub-bloques [.1 - .64] y [.65 - .127].

De los dos bloques, uno de ellos es el que contiene la dirección IP del nodo que facilita la entrada al nuevo (la dirección IP .65 está en el bloque [.65 - .127]).

En este caso, al contrario de lo que ha sucedido antes, se entregará al nuevo nodo el bloque con las direcciones más bajas, para evitar que la dirección IP .65 sea usada por cualquier otro nodo ya que en realidad no está libre (véase Tabla 6.10).

Tabla 6.9: Tabla Free_IP_Blocks al ingresar el nodo .128 a la red

| IP | Bloque de Direcciones |
|---|---|
| .65 | .1 - .127 |
| .128 | .128 - .254 |

Tabla 6.10: Tabla Free_IP_Blocks al ingresar el nodo .1 a la red

| IP | Bloque de Direcciones |
|---|---|
| .65 | .65 - .127 |
| .128 | .128 - .254 |
| .1 | .1 - .64 |

### 6.5.2 Escenarios Problemáticos

Al realizar la actualización de la tabla de autoconfiguración cada nodo de forma local, sin intercambiar mensajes que confirmen los cambios, debe tenerse la certeza de que ninguna situación pueda desencadenar que un nodo tenga almacenada información que no se corresponda con la situación real de la red. Existen varias situaciones conflictivas con el modelo de sincronización propuesto:

- Un nodo perteneciente a la red ofrece la mitad de su bloque de direcciones IP libres a los nodos que intentan acceder. Se debe tener en cuenta que tras ofrecer esas direcciones IP, el nodo servidor puede recibir otra solicitud diferente. En ese caso, el nodo no debe responder hasta tener confirmación de que el ofrecimiento anterior ha sido aceptado o rechazado.

  Conocer si el nodo cliente al que se le han ofrecido las direcciones IP libres ha aceptado la solicitud es simple, ya que el protocolo OLSR detectará una nueva ruta hacia un nodo con la primera dirección IP del bloque ofrecido.

  Para detectar si una solicitud no ha sido aceptada, el nodo servidor pone en marcha un temporizador desde el momento en que se envía el primer ofrecimiento. Si al expirar este temporizador no se tiene constancia de que el nodo cliente ha aceptado el ofrecimiento, esto significa que ha sido rechazado, ya que la oferta habrá caducado y el nodo cliente no podrá usar las direcciones ofrecidas.

- Otro escenario problemático al que se debe prestar atención es la detección de entradas o salidas de nodos fuera de orden. Para explicar este escenario se desarrolla a continuación un ejemplo práctico.

  Supóngase que la Tabla 6.11 representa parte de una red en tres instantes de tiempo diferentes: $t_1$, $t_2$ y $t_3$. Se representan tres nodos y el bloque de direcciones que corresponde a cada nodo en los diferentes instantes. En un primer momento el único

nodo de la red representada es el nodo A. Más adelante se incorporan a ésta el nodo B y luego C, siendo en ambos casos el nodo A quien les facilita la entrada.

Tabla 6.11: Fracción del estado de una red en tres instantes diferentes

| **Nodo** | **IP** | $\mathbf{t_1}$ | $\mathbf{t_2}$ | $\mathbf{t_3}$ |
|---|---|---|---|---|
| A | .1 | .1 - .100 | .1 - .50 | .1 - .25 |
| B | .51 | | .51 - .100 | .51 - .100 |
| C | .26 | | | .26 - .50 |

Si ésta es parte de una red más grande, estos cambios son detectados por otro nodo perteneciente a ella, aquí no representado. En su tabla de autoconfiguración tiene almacenada la información de que el nodo A usa la dirección IP .1 y el bloque de direcciones IP libres [.1 - .100]. Mediante la monitorización de la tabla de encaminamiento del protocolo OLSR detectará la entrada de los nodos B (.51) y C (.26).

Pero, por problemas de interferencias, saturación de la red o pérdida de mensajes, puede ocurrir que OLSR incluya antes una ruta hacia el nodo C que hacia el nodo B. Desde el punto de vista de la autoconfiguración lo que pasa es que en el instante $t_2$ aparece un nodo con la dirección IP .26. Esto implica que esa dirección ha sido facilitada por A, ya que tenía asignado el bloque de direcciones [.1 - .100], lo cual es incorrecto. A habría dividido su bloque de direcciones en dos sub-bloques, [.1 - .50] y [.51 - .100], por lo que la aparición de un nodo con la dirección IP .26 no es posible (véase Tabla 6.12).

Tabla 6.12: Fracción del estado de una red al ingresar el nodo .26

| **Nodo** | **IP** | $\mathbf{t_1}$ | $\mathbf{t_2}$ | $\mathbf{t_3}$ |
|---|---|---|---|---|
| A | .1 | .1 - .100 | .1 - .50 | |
| B | | | | |
| C | .26 | | .51 - .100 | |

- Un problema muy similar a éste se da si un nodo abandona la red en un momento cercano en el que el servidor que debe recoger las direcciones que quedan disponibles está dando entrada a un nuevo cliente. Otro nodo de la red podría detectar la nueva entrada y la salida del nodo de manera que la tabla de autoconfiguración resultante no tuviese sentido.

  La solución propuesta consiste en establecer temporizadores, de manera que se garantice que la entrada de un nuevo nodo en la red no esté cercana a otra entrada o salida, que modifique los bloques de direcciones IP involucrados en esa nueva entrada.

Tras otorgar un bloque de direcciones IP libres a un nuevo nodo cliente, el servidor establece un tiempo durante el cual no atenderá más peticiones. Asimismo, el nodo que acaba de entrar en la red esperará el mismo tiempo para comenzar a atender peticiones de otros nodos.

Siguiendo con el ejemplo anterior, tras facilitar la entrada al nodo B, el nodo A no atendería la solicitud de C hasta que pasase un tiempo suficiente como para que cualquier nodo de la red haya podido detectar la nueva presencia de B. Por tanto, no existe riesgo de que la entrada del nodo C sea detectada antes que la de B por algún otro nodo.

De manera similar, cuando un nodo detecta que debe recoger las direcciones que un nodo que acaba de abandonar la red deja disponibles, espera un tiempo suficiente antes de volver a atender peticiones de clientes. Así, el resto de la red actualizará el bloque de direcciones correspondiente antes de que se divida y se entregue la mitad a algún cliente.

## 6.6 Inicialización

Se ha visto que se actualiza la información necesaria para sincronizar el protocolo D2HCP aprovechando las actualizaciones de rutas de OLSR. Pero para poder actualizar la tabla un nodo que acaba de entrar a la red debe primero construirla.

Cuando un nuevo nodo accede a la red, pero todavía no pertenece a ésta, esto es, durante el proceso de configuración, el nodo servidor transmite una copia de su tabla de autoconfiguración al nodo cliente. El mensaje *IP_Assigned* contiene la tabla mencionada. En otras palabras, desde el momento en que un nodo comienza a participar en la red dispone de una tabla de autoconfiguración que puede actualizar con la información proporcionada por OLSR.

Hay que aclarar que aunque la información sobre los bloques de direcciones libres y la topología de la red están relacionadas, en este primer momento se dispone de la tabla de autoconfiguración. OLSR comienza a funcionar de aquí en adelante. La *tabla de encaminamiento* se encuentra vacía.

Si OLSR detecta una ruta nueva hacia un nodo antes desconocido, se actualizará además la tabla de autoconfiguración. Si ese nodo que OLSR descubre como nuevo se encuentra en la tabla que se recibió del nodo servidor, no habrá que actualizar ninguna entrada.

Se puede dar el caso de que tras recibir esa tabla de autoconfiguración con todos los nodos de la red, se produzca la salida de uno de ellos. Como se ha dicho, durante un tiempo inicial OLSR no contiene ninguna ruta y las irá descubriendo desde ese momento. Por tanto, desde el punto de vista de OLSR, el que un nodo desaparezca de la red en esos primeros instantes no implica borrar una ruta, ya que no se disponía de ninguna. En este caso la detección de esa salida se realiza mediante un temporizador.

Al recibir la tabla se inicia un temporizador. Cuando éste expira, se borrarán de la tabla de autoconfiguración los nodos para los cuales OLSR no tenga ruta conocida. De este modo, el problema que se daría porque un nodo abandone la red sin que OLSR lo detecte queda resuelto.

Tras cubrir cómo se resolvieron la sincronización y la inicialización, se observa que el protocolo OLSR es monitorizado, pero no se modifica ni su comportamiento ni sus mensajes. Esto implica que la solución propuesta genera una **sobrecarga nula** sobre el tráfico originado por OLSR una vez que el nuevo nodo esté configurado correctamente.

## 6.7 Formato de los Mensajes

Todos los mensajes enviados en el protocolo van empaquetados con el formato presentado en la Figura 6.2. Estos mensajes serán encapsulados a su vez con las cabeceras correspondientes a la capa MAC o TCP/IP, dependiendo del tipo de mensaje que se incluya en el campo *Message*.

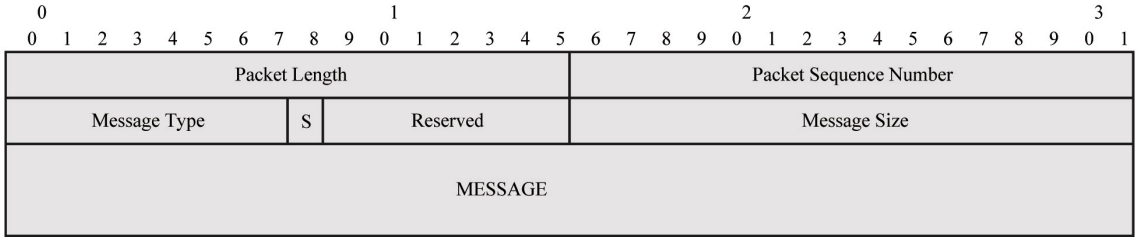



Figura 6.2: Paquete del protocolo D2HCP

### 6.7.1 Cabecera del Paquete

La primera fila de la Figura 6.2 contiene los campos de la cabecera del paquete:

- **Packet Length**: Longitud del paquete, incluyendo la cabecera (2 bytes).
- **Packet Sequence Number**: Número de secuencia (2 bytes). En cada mensaje diferente que envía un nodo este campo se incrementa en uno. Sirve para poder detectar paquetes duplicados.

### 6.7.2 Cabecera de los Mensajes

La segunda fila de la Figura 6.2 constituye la cabecera para cada uno de los mensajes del protocolo:

- **Message Type**: Tipo de mensaje (1 byte).
- **S (Security)**: Reservado para la implementación de seguridad (1 bit).
- **Reserved**: Reservado para futuras ampliaciones de funcionalidad (7 bits).
- **Message Size**: Tamaño del mensaje (2 bytes).

A continuación se muestra el formato de cada tipo de mensaje, contenido en el campo *Message*.

### 6.7.3 Server_Discovery

La Figura 6.3 muestra el formato del mensaje *Server_Discovery*.
Los campos son los siguientes:

- **ID**: Identificador del nodo (6 bytes). Si el nodo tiene más de una interfaz, debe elegir la dirección MAC de una de ellas. Este campo tiene el mismo valor para cualquier mensaje *Server_Discovery* / *Server_Poll* emitido por el nodo aunque se haga desde distintas interfaces.

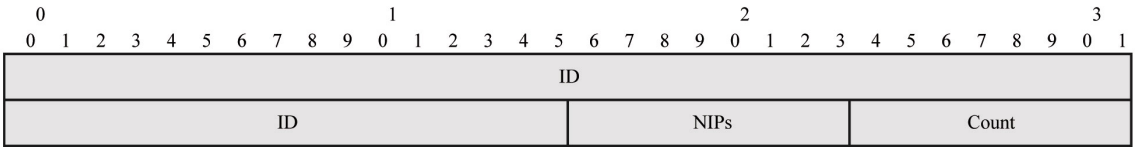



Figura 6.3: Mensaje Server_Discovery

- **NIPs**: Número de direcciones IP solicitadas por el nodo. Será igual al número de interfaces del nodo cliente (1 byte).
- **Count**: Número de veces que se ha intentado la petición *Server_Discovery* (1 byte).

### 6.7.4 Server_Offer

La Figura 6.4 muestra el formato del mensaje *Server_Offer*.

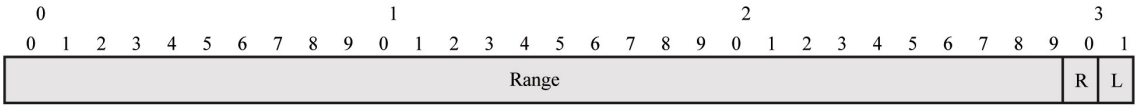



Figura 6.4: Mensaje Server_Offer

Los campos son los siguientes:

- **Range**: Número de direcciones IP que se ofrecen (30 bits).
- **Ready (R)**: Indica si el nodo que ofrece las direcciones IP está listo para asignarlas de inmediato o si tan sólo comunica su existencia aunque en este momento no esté en disposición de asignarlas (1 bit).
- **Local (L)**: Indica si el rango ofrecido es del nodo emisor o, por el contrario, se pedirá a su vez a un tercer nodo de la red (1 bit).

### 6.7.5 Server_Poll

La Figura 6.5 muestra el formato del mensaje *Server_Poll*.

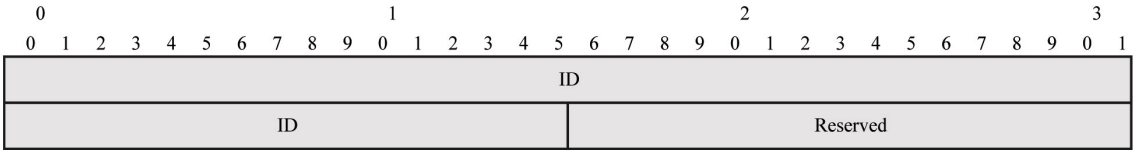



Figura 6.5: Mensaje Server_Poll

Los campos son los siguientes:

- **ID**: Identificador del nodo (6 bytes) Es el mismo identificador que el elegido en el mensaje *Server_Discovery*.

- **Reserved**: Reservado para futuras implementaciones (2 bytes).

### 6.7.6 IP_Assigned

La Figura 6.6 muestra el formato del mensaje *IP_Assigned*.

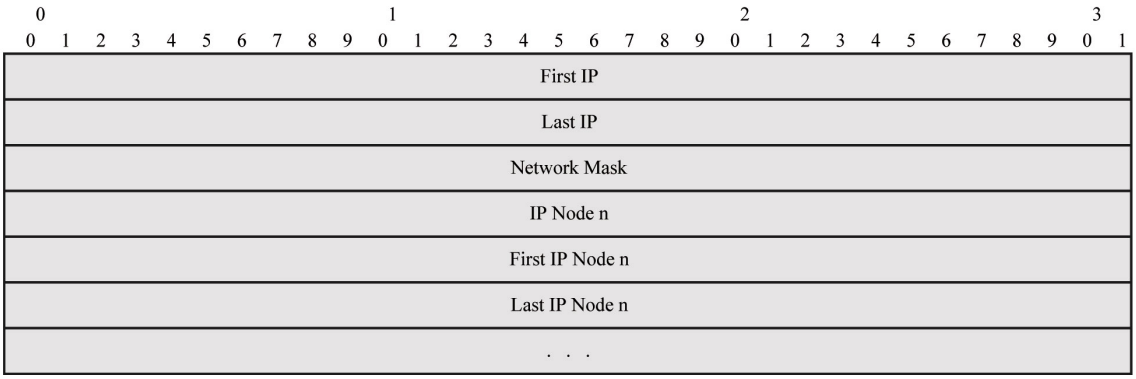



Figura 6.6: Mensaje IP_Assigned

Los campos son los siguientes:

- **First IP**: Dirección IP inicial del bloque de direcciones libres (4 bytes).
- **Last IP**: Dirección IP final del bloque de direcciones libres (4 bytes).
- **Network Mask**: Máscara de red (4 bytes).
- **IP Node n, First IP Node n, Last IP Node n**: Representa una entrada de la tabla de bloques libres de todos los nodos de la red. Cada nodo está representado por 3 campos de 4 bytes: la dirección IP del nodo y las direcciones IP inicial y final de su bloque de direcciones libres.

### 6.7.7 IP_Range_Request

La Figura 6.7 muestra el formato del mensaje *IP_Range_Request*.

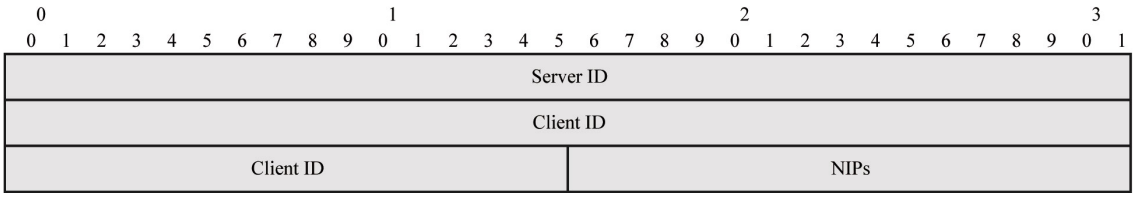



Figura 6.7: Mensaje IP_Range_Request

Los campos son los siguientes:

- **Server IP**: Dirección del servidor que solicita el rango de direcciones para el cliente (4 bytes). Al tratarse de redes multisalto puede ser diferente de la dirección IP del emisor del mensaje.

- **Client ID**: Identificador del nodo cliente. Tiene el mismo valor que el campo ID de los mensajes *Server_Discovery* / *Server_Poll* (6 bytes).
- **NIPs**: Número de direcciones IP solicitadas (2 bytes).

### 6.7.8 IP_Range_Return

La Figura 6.8 muestra el formato del mensaje *IP_Range_Return*.

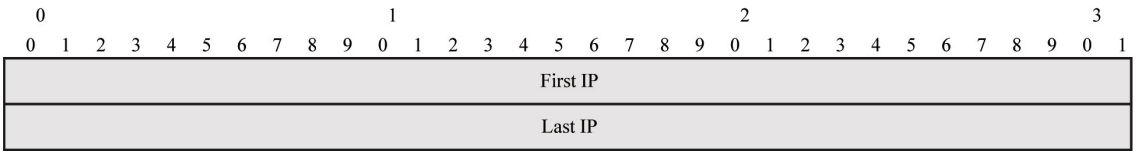



Figura 6.8: Mensaje IP_Range_Return

Los campos son los siguientes:

- **First IP**: Dirección IP inicial del bloque de direcciones libres (4 bytes).
- **Last IP**: Dirección IP final del bloque de direcciones libres (4 bytes).

## 6.8 Temporizadores

El carácter inalámbrico y móvil de las redes móviles ad hoc provoca que en este tipo de redes se den con frecuencia situaciones en las que se pierdan mensajes o tarden en llegar a su destino más de lo estimado. El protocolo D2HCP utiliza 10 temporizadores para resolver posibles situaciones de ese tipo. Estos temporizadores son:

- *Server_Discovery_Timer*
- *Server_Offer_Timer*
- *Server_Poll_Timer*
- *IP_Range_Request_Timer*
- *Accepted_Offer_Timer*
- *Node_Down_Timer*
- *Init_Table_Timer*
- *Init_Assign_Timer*
- *Node_Down_Assign_Timer*
- *Sleep_Timer*

### 6.8.1 Server_Discovery_Timer

Tras enviar el mensaje *Server_Discovery*, el nodo cliente inicia este temporizador al pasar al estado *Waiting_Reply*. Durante este tiempo el nodo está esperando mensajes *Server_Offer* de los posibles nodos cercanos de la red. Cuanto mayor sea el temporizador mayor será el tiempo dedicado a recibir mensajes de este tipo, por lo que habrá más mensajes para procesar y, por tanto, más fácil será obtener un bloque de direcciones. No obstante, la latencia para obtener una dirección IP también será mayor.

Si este temporizador expira y el nodo cliente no ha recibido ningún *Server_Offer*, bien por pérdidas de mensajes bien porque no hay ningún nodo servidor, vuelve a enviar un nuevo *Server_Discovery*. Esta acción se repite un número máximo (*Sdiscovery_Max_Retry*) de veces, transcurridos los cuales, sin recibir mensajes, inicia su propia red.

### 6.8.2 Server_Offer_Timer

Tras el mensaje *Server_Offer* el nodo pasa al estado *Waiting_Poll* e inicia este temporizador. Al expirar éste, el estado cambia a IDLE.

### 6.8.3 Server_Poll_Timer

Una vez enviado el mensaje *Server_Poll*, el nodo cliente queda a la espera de recibir el mensaje *IP_Assigned* durante el tiempo que determine este temporizador. Si este mensaje no llega, se retransmite el *Server_Poll* hasta un número máximo (*Spoll_Max_Retry*) de intentos. Si se supera el número máximo de intentos, comienza el proceso de configuración de nuevo.

### 6.8.4 IP_Range_Request_Timer

Cuando un nodo servidor envía un mensaje *IP_Range_Request* a otro nodo de la red inicia este temporizador. Al igual que con el *Server_Poll_Timer*, si expira este temporizador se reenvía el mensaje *IP_Range_Request* hasta un número máximo (*Rrequest_Max_Retry*) de intentos.

### 6.8.5 Accepted_Offer_Timer

Este temporizador se activa después de enviar un mensaje *IP_Assigned* o un mensaje *IP_Range_Return*. Durante este tiempo el nodo servidor no puede responder a solicitudes de tipo *Server_Poll* o *IP_Range_Request*. Esta restricción se levanta al expirar el temporizador (la oferta caducó sin ser aceptada) o al detectar que un nodo con la primera dirección IP de las ofrecidas ha entrado en la red (el bloque de direcciones ofrecido fue aceptado).

Hay que tener en cuenta que aunque no pueda asignar direcciones IP, el nodo servidor seguirá respondiendo a peticiones *Server_Discovery* con el mensaje *Server_Offer* (R = 0). De este modo se informa al nodo cliente de la existencia del servidor, aunque no sea capaz de asignar direcciones IP inmediatamente.

### 6.8.6 Node_Down_Timer

Cuando OLSR borra la ruta hacia un nodo, no se elimina inmediatamente de la tabla *Free_IP_Blocks*. En su lugar se inicia este temporizador. Si antes de que el temporizador expire se vuelve a descubrir una ruta hacia el nodo, éste desapareció momentáneamente, pero no abandonó la red. Por tanto, se cancela la eliminación de esa entrada en la tabla

*Free_IP_Blocks*. En el caso de que el temporizador expire y no se haya recuperado la ruta, se da al nodo por perdido y se elimina su entrada de la tabla *Free_IP_Blocks*, actualizando las que correspondan.

### 6.8.7 Init_Table_Timer

El nodo cliente activa este temporizador cuando recibe la tabla de autoconfiguración *Free_IP_Blocks* en el mensaje *IP_Assigned*. Durante este tiempo la tabla contiene nodos para los que OLSR aún no tiene una ruta conocida. Al expirar el temporizador se comprueba qué nodos de la tabla *Free_IP_Blocks* no tienen entrada en la *tabla de encaminamiento* de OLSR eliminándose (actualizando las entradas que correspondan), ya que son nodos que pertenecían a la red al recibir la tabla y la han abandonado antes de que OLSR supiese de su existencia.

### 6.8.8 Init_Assign_Timer

Este temporizador lo usa tanto el nodo cliente como el servidor cuando han recibido o asignado, respectivamente, un bloque de direcciones IP. Es decir, el servidor lo inicializa al comprobar la entrada de un nodo con la primera dirección IP del bloque ofrecido en el mensaje *IP_Assigned* y el cliente lo inicia tras recibir el mensaje *IP_Assigned* y configurar su dirección. Así se da tiempo a que toda la red pueda actualizar su tabla *Free_IP_Blocks* antes de que se produzcan más cambios. Durante este tiempo se ignoran mensajes *Server_Poll* o *IP_Range_Request*, aunque se responden a los *Server_Discovery*.

### 6.8.9 Node_Down_Assign_Timer

Se pone en marcha este temporizador cuando un nodo ya configurado detecta la salida de otro y comprueba que le corresponde recoger las direcciones IP que quedan libres. Más concretamente, el temporizador se activa cuando se detecta la eliminación de esa entrada en la tabla de encaminamiento OLSR, es decir, se activa al mismo tiempo que el temporizador *Node_Down_Timer*.

Hasta que no expire este temporizador, el nodo ignora las peticiones *Server_Poll* e *IP_Range_Request*. De este modo se da un margen de tiempo para asegurarse de que todos los nodos en la red detectan la salida mencionada y actualizan su tabla *Free_IP_Blocks*, antes de asignarlas a algún nuevo nodo. Por tanto, la duración de este temporizador debe ser mayor que la del *Node_Down_Timer* para asegurarse de que el resto de nodos de la red no sólo han detectado la eliminación de una ruta sino que la han eliminado de la tabla *Free_IP_Blocks*. El nodo sigue respondiendo a los mensajes *Server_Discovery* con el mensaje *Server_Offer* con (R = 0).

### 6.8.10 Sleep_Timer

Temporizador usado por un nodo cliente cuando detecta nodos cercanos pertenecientes a alguna red, pero que no están en disposición de asignar direcciones IP en ese momento. De este modo se da un margen de tiempo para permitir que acaben los procesos que les impiden asignar direcciones.

## 6.9 Diagramas de Estado

Dependiendo de si están en proceso de entrar en la red o de si ya pertenecen a ésta, se distinguen dos tipos de nodos: nodo cliente y nodo servidor. En los siguientes apartados

se muestran y explican los diagramas de estados que rigen el comportamiento de ambos tipos de nodos. El estado en el que se encuentra un nodo cambia al producirse envíos o recepciones de mensajes, o cuando determinados temporizadores expiran.

### 6.9.1 Nodo Servidor

Se denomina nodo servidor a todos los nodos de la red que están correctamente configurados, es decir, que poseen una dirección IP válida con la que se comunican con el resto de nodos y un bloque de direcciones IP libres. Con este último facilitan el acceso a los nuevos nodos, que se denominan nodos clientes.

El diagrama de estados de un nodo servidor se muestra en la Figura 6.9.

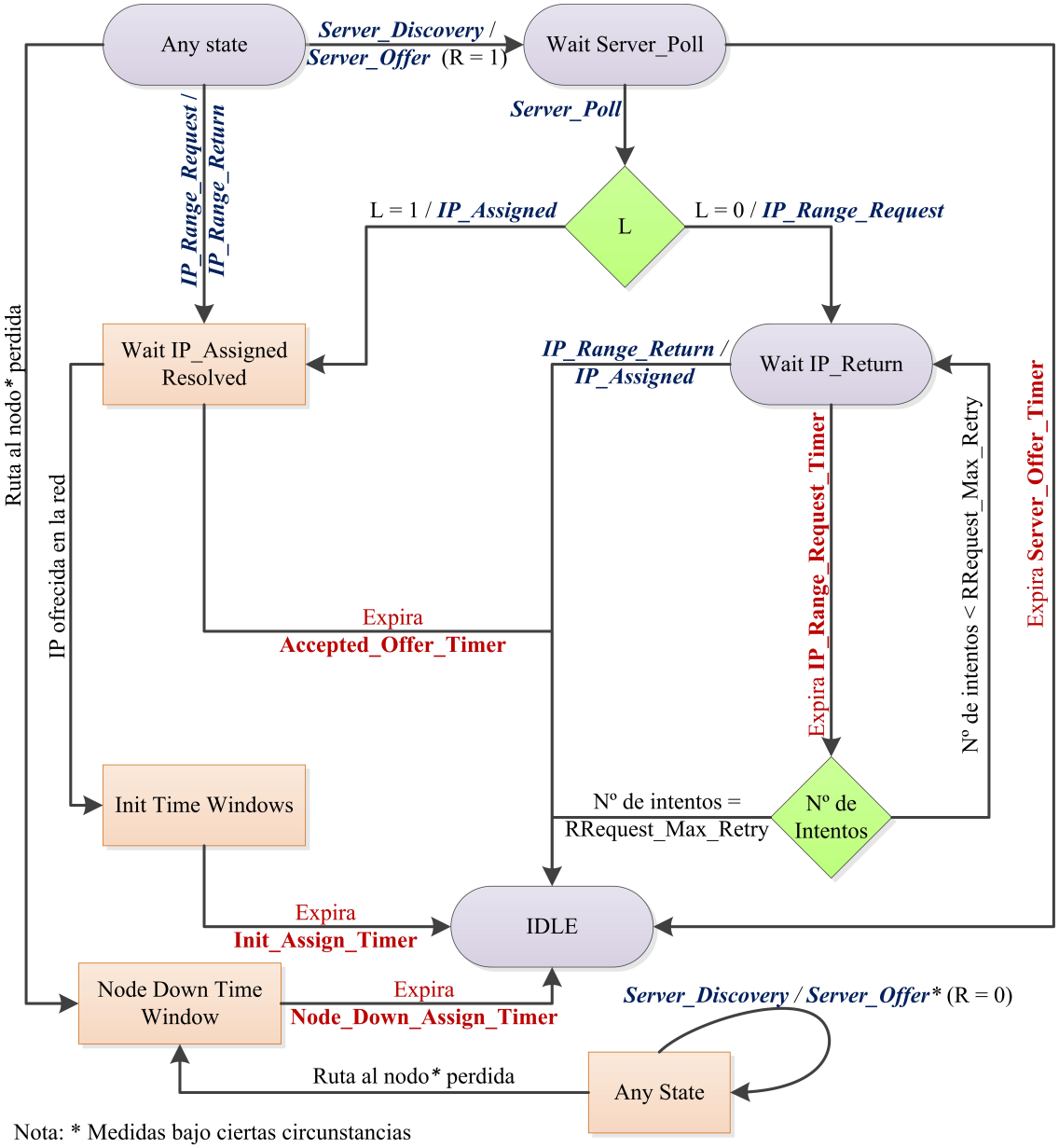



Figura 6.9: Diagrama de estados del nodo servidor

Como se observa en la citada Figura, existen dos tipos de estados: los representados con rectángulos redondeados y los que se encierran en rectángulos con esquinas. Se denominan estados tipo *ready* o *not_ready*, respectivamente.

En todos los estados, el nodo servidor está esperando mensajes *Server_Discovery*. Responde siempre con un mensaje *Server_Offer*. Dependiendo de si el estado actual del nodo es de tipo *ready* o *not_ready*, responde con R = 1 o R = 0, respectivamente. De este modo un nodo siempre anuncia su presencia, incluso cuando no sea capaz de asignar direcciones IP a un cliente. Esto se indica en el diagrama de estados con los estados *Any state*. Estos estados son:

- *Any state (ready)*
- *Any state (not_ready)*
- *Wait Server_Poll*
- *Wait IP_Assigned Resolved*
- *Init Time Window*
- *Wait IP_Return*
- *IDLE*
- *Node Down Time Window*

A continuación se comenta brevemente el significado de cada uno de los estados:

- **Any state (ready)**: En cualquier estado *ready* el nodo servidor responde a un mensaje *Server_Discovery* con uno del tipo *Server_Offer* con R = 1. Esto hace que el servidor pase al estado *Wait Server_Poll*.

  Un nodo puede responder al mismo tiempo peticiones *Server_Discovery* de nodos diferentes y estar a la espera de cualquiera de los correspondientes *Server_Poll*.

  También se responden a mensajes del tipo *IP_Range_Request* con mensajes del tipo *IP_Range_Return*. Esto significa que si el nodo se encuentra en cualquier estado *ready*, da la mitad de su bloque de direcciones libres a cualquier otro nodo de la red sin direcciones propias que necesite facilitar la entrada a un cliente.

- **Any state (not_ready)**: Mientras el nodo se encuentre en un estado *not_ready*, responde a los mensajes *Server_Discovery* con un mensaje *Server_Offer* con R = 0.

- **Wait Server_Poll**: En este estado el nodo espera un tiempo determinado por el temporizador *Server_Offer_Timer* la recepción de un mensaje *Server_Poll*. Este mensaje indica que el cliente a quien envió el *Server_Offer* le ha elegido como su servidor para el proceso de autoconfiguración.

  Tras la recepción del mensaje, el nodo envía un mensaje *IP_Assigned* al cliente en caso de tener direcciones disponibles localmente (*Server_Offer* con L = 1) pasando al estado *Wait IP_Assigned Resolved*.

  Si las direcciones que ofreció no son locales, debe pedirlas a un nodo de la red con el mensaje *IP_Range_Request* pasando al estado *Wait IP_RReturn*.

  En caso de que no se reciba ningún mensaje *Server_Offer* antes de que el temporizador expire, pasa al estado IDLE.

- **Wait IP_Assigned Resolved**: En este estado *not_ready* el nodo servidor espera mientras conoce si el nodo cliente recibió correctamente el bloque de direcciones ofrecido mediante un mensaje *IP_Assigned* o a través de un mensaje *IP_Range_Return* y un intermediario. El resultado puede ser que el cliente se haya configurado correctamente o que no haya recibido el bloque de direcciones. Esto último puede ocurrir por varios motivos (problemas de interferencias en la recepción del mensaje, movimiento del nodo cliente fuera del rango de cobertura, etc.).

  Si aparece un nuevo nodo en la red usando la primera dirección IP del bloque ofrecido en el mensaje *IP_Assigned* o *IP_Range_Return*, el nodo cliente terminó de configurarse y el nodo servidor cambia su estado a *Init time window*.

  Si el nodo no fue capaz de terminar el proceso de autoconfiguración, el temporizador *Accepted_Offer_Timer* expira y el nodo pasa al estado IDLE.

- **Init Time Window**: Este estado sirve para dar un margen de tiempo que permita a todos los nodos de la red detectar la entrada del cliente recientemente configurado, antes de volver a dividir el propio bloque de direcciones IP libres.

  Si no se diese este margen y se atendiesen peticiones nuevas de inmediato, podrían darse problemas de sincronización si otros nodos detectan las nuevas incorporaciones a la red en un orden incorrecto.

- **Wait IP_Return**: En este estado el nodo está esperando recibir un *IP_Range_Return*.

  Si recibe el citado mensaje, en el que un nodo de la red le indica un bloque que puede ofrecer al cliente en espera, envía al cliente un mensaje *IP_Assigned*. Tras esto, el nodo habrá terminado su función como servidor pasando al estado IDLE.

  Si no lo recibe antes de que expire el temporizador *IP_Range_Request_Timer*, se vuelve a intentar la petición enviando de nuevo un mensaje *IP_Range_Request* un número máximo (*Rrequest_Max_Retry*) de veces. Estos sucesivos intentos se envían en cada ocasión a un nodo diferente. Si se sobrepasa el límite de intentos entonces el nodo desiste y cambia su estado a IDLE.

- **IDLE**: Estado de reposo durante el cual el nodo está ocioso. Cuando se encuentra en este estado, el nodo no está realizando ninguna operación relacionada con la autoconfiguración. Se trata, por tanto, de un estado de espera.

- **Node Down Time Window**: Este nodo proporciona un margen de tiempo cuando el nodo debe recoger las direcciones IP de otro nodo que ha abandonado la red.

  Más exactamente, el nodo cambia a este estado al detectar que ha perdido la ruta hacia un nodo de cuyo bloque de direcciones es responsable. Esta transición se hace desde cualquier otro estado, *ready* o *not_ready*.

  Tras el tiempo determinado por el temporizador *Node_Down_Assign_Timer*, el nodo vuelve al estado de reposo IDLE.

  No debe confundirse este temporizador con el *Node_Down_Timer*. Aunque se inicien al mismo tiempo al detectar el mismo evento, los procesos involucrados son independientes.

### 6.9.2 Nodo Cliente

El procedimiento que sigue un nodo que desea acceder a una red se describe en la Figura 6.10. El diagrama de estados cuenta también con estados *not_ready*.

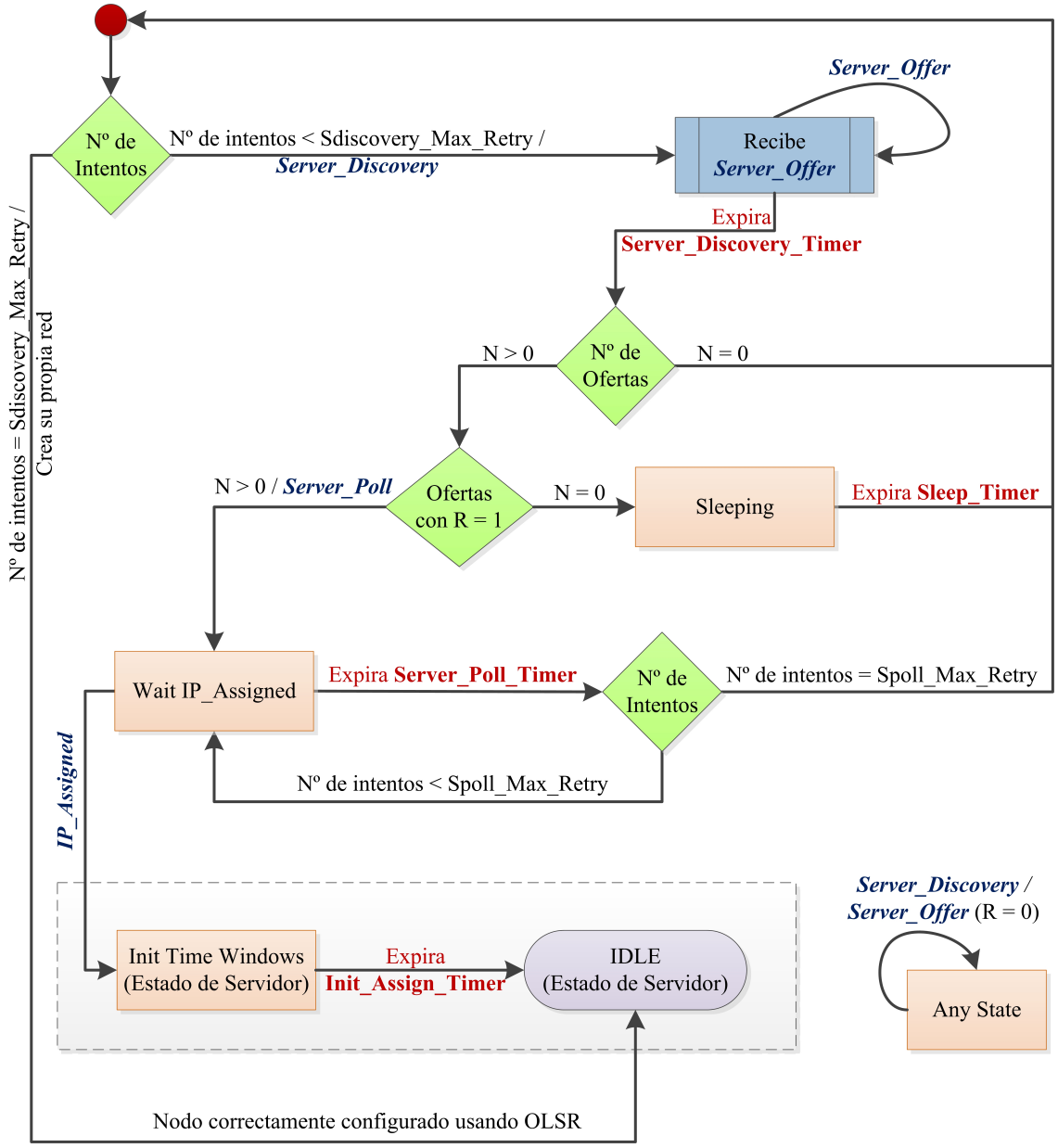



Figura 6.10: Diagrama de estados del nodo cliente

Al igual que si se tratara de un nodo ya configurado, el nodo que está en proceso de configuración de su dirección IP responde a peticiones *Server_Discovery* con mensajes *Server_Offer*. En estos mensajes R = 0, ya que el nodo no está en disposición de asignar direcciones IP. Sólo pretende anunciar su presencia.

Estos estados son:

- *Initial state*
- *Receive Server_Offer*
- *Sleeping*
- *Wait IP_Assigned*

A continuación se comenta brevemente el significado de cada uno de los estados:

- **Initial state**: Estado inicial en el que comienza el proceso de autoconfiguración.

  Si el número de intentos es menor del máximo (*Sdiscovery_Max_Retry*) entonces el nodo emite un mensaje *Server_Discovery* a través de cada una de las interfaces de red que vaya a utilizar en la red móvil ad hoc, cambiando su estado a *Receive Server_Offer*.

  Si se ha llegado al límite de intentos, entonces el nodo desiste de su intención de encontrar una red a la que unirse y crea una nueva. Pasa a ser un nodo servidor en el estado IDLE del diagrama de estados del servidor.

- **Receive Server_Offer**: Se trata de un estado de espera durante el cual se recogen los mensajes *Server_Offer* de posibles nodos próximos. Al terminar la espera, determinada por el temporizador *Server_Discovery_Timer*, se procesan las respuestas:

  - Si no se ha recibido ninguna respuesta *Server_Offer* se vuelve al estado inicial.
  - Si hay alguna oferta se comprueba el número de ellas con R = 1.
  - Si de entre las ofertas ninguna tiene R = 1, hay nodos cercanos pertenecientes a una red, pero de momento no son capaces de asignar direcciones IP, pasando el nodo al estado *Sleeping*.
  - Si hay alguna oferta con R = 1 se ordenan los servidores por preferencia y se envía un mensaje *Server_Poll* al primero de ellos, cambiando su estado a *Wait IP_Assigned*.

  Este estado no es de ninguno de los tipos explicados en el diagrama de estados. No responde a mensajes *Server_Discovery* ya que de momento no conoce si hay alguna red cercana a la que puede unirse.

- **Sleeping**: El nodo pausa sus intentos de entrar en la red durante el tiempo determinado por el temporizador *Sleep_Timer*. Se han recibido mensajes *Server_Offer* de nodos cercanos que de momento no son capaces de asignar direcciones IP y se pretende que tras este tiempo ya sean capaces de facilitar la entrada a la red.

  Al expirar el temporizador, el nodo vuelve al estado inicial.

- **Wait IP_Assigned**: El cliente se encuentra esperando un mensaje *IP_Assigned* por parte del servidor al que se envió el mensaje *Server_Poll*.

  Si no llega el mensaje esperado, el temporizador *Server_Poll_Timer* expira. En este caso, se envía un *Server_Poll* al siguiente servidor de la lista generada tras la finalización del temporizador *Server_Discovery_Timer*. Si se acaba la lista de servidores o se llega al límite de intentos *Spoll_Max_Retry*, el nodo vuelve al estado inicial.

  Al recibir el mensaje *IP_Assigned*, el nodo configura su dirección (o direcciones, en caso de disponer de varias interfaces de red). A partir de entonces participa en la red pasando a ser un nodo servidor en el estado *Init time window*.

## 6.10 Síntesis del Capítulo

El principal objetivo de este capítulo ha sido la especificación del protocolo de autoconfiguración para redes móviles ad hoc denominado D2HCP. Se han enunciado las

características del citado protocolo así como sus diferencias respecto a su predecesor, el protocolo de Mohsin y Prakash. Se ha descrito la entrada y salida de nodos, analizando la alta disponibilidad del protocolo durante la entrada de nodos, así como su simplicidad en la salida. Asimismo, se ha prestado especial atención a la gestión de la estructuras de datos del protocolo y a su sistema de sincronización que, al apoyarse en el protocolo de encaminamiento OLSR, genera una sobrecarga nula en el tráfico de la red con respecto al generado por el protocolo OLSR. Estos dos aspectos en sí mismos constituyen dos de las aportaciones más relevantes de este trabajo. Por último, se han analizado los mensajes y los temporizadores de D2HCP, así como el diagrama de estado de los nodos cliente y servidor.

# Capítulo 7

# Protocolo E-D2HCP

Este capítulo presenta el diseño y especificación de una extensión del protocolo de autoconfiguración para redes móviles ad hoc D2HCP presentado en el capítulo anterior. La extensión denominada *Protocolo Mejorado de Configuración de Host Dinámico y Distribuido* o, más comúnmente, E-D2HCP, como se cita en lo sucesivo, expresión que corresponde al acrónimo de su denominación inglesa *Enhanced Distributed Dynamic Host Configuration Protocol*. El capítulo comienza describiendo las principales características del protocolo E-D2HCP. Después, analiza las principales diferencias respecto a su predecesor. A continuación se detallan las estructuras de datos usadas por el protocolo. Seguidamente se analizan la entrada y salida de nodos, el proceso de sincronización del protocolo, la inicialización de la red y la fusión de redes. Posteriormente, se proporciona una completa especificación del protocolo, detallando los mensajes, los temporizadores y el diagrama de estado de los nodos cliente y servidor en E-D2HCP. El capítulo finaliza con una breve síntesis de lo expuesto en el mismo.

## 7.1 Generalidades

Al igual que su predecesor, E-D2HCP es un protocolo de autoconfiguración que gestiona la entrada y salida de nodos en redes móviles ad hoc.

El protocolo hace que los nodos de una red móvil ad hoc colaboren entre sí para gestionar la asignación de direcciones IP únicas y correctas de forma distribuida. Todos los nodos de la red tienen el mismo papel, no existiendo un tipo de nodo especial que centralice la gestión de la misma.

Los nodos cuentan con un sistema de sincronización que se apoya en el protocolo de encaminamiento OLSR. Gracias a este mecanismo la sincronización se lleva a cabo de forma pasiva, monitorizando el protocolo de encaminamiento mencionado, por lo que se genera una sobrecarga nula en el tráfico de la red con respecto al generado por el protocolo OLSR.

Al ser todos los nodos responsables de gestionar la entrada de cualquier otro nuevo nodo a la red, esta operación puede realizarse rápidamente. Un nodo que desea entrar a una red intenta contactar con cualquier nodo ya perteneciente a ella, pudiendo recibir respuesta de varios nodos. Esto hace que las posibilidades de entrar a formar parte de la red con éxito sean altas, debido a la alta disponibilidad y a la redundancia que supone la gestión distribuida.

Además de las características anteriores que comparte con su predecesor, E-D2HCP presenta otras avanzadas, las más importantes son:

- Soporta la fusión de redes.
- Presenta una mejor tolerancia a fallos (caídas) de los enlaces.
- Mejora la inicialización de la red, particularmente en el caso de que dos nodos quieran iniciar la red concurrentemente.

Las 3 características anteriores determinan que métricas (como la latencia y la sobrecarga) mejoren ostensiblemente.

## 7.2 Consideraciones de Diseño

Básicamente, E-D2HCP es una extensión de D2HCP, derivándose de este último mediante los siguientes cambios:

1. Se han añadido 4 nuevos mensajes:
   - **Client_Synchronization**: Este mensaje se envía cuando dos nodos adyacentes empiezan el proceso de autoconfiguración para establecer cuál de ellos tendrá prioridad para iniciar la red.
   - **Hello**: Este mensaje se envía periódicamente para detectar posibles fusiones de redes.
   - **Merge_Network** e **IP_Address_Invalidate**: Se envían estos mensajes cuando se detecta una fusión para redistribuir las direcciones IP en la red resultante.
2. Se han modificado varios estados:
   - **SearchingServer**: Indica que el nodo ha enviado el mensaje *Server_Discovery*.
   - **Waiting_Allocation**: Indica que el nodo cliente ha encontrado un servidor que puede proporcionarle una dirección válida.
   - **Suspended**: Un nodo cliente pasa a este estado cuando ninguno de los nodos que han contestado al mensaje *Server_Discovery* ha completado el proceso de configuración.
   - **Synchronizing**: Cuando dos nodos adyacentes no han encontrado ningún nodo que haya comenzado o terminado el proceso de configuración construyen una nueva red. En el estado *Synchronizing* se toma la decisión de qué nodo tendrá prioridad en este proceso.
3. Además de las características anteriores, que pueden considerarse como las más relevantes, se han realizado algunas modificaciones a nivel del formato de los mensajes del protocolo.

## 7.3 Estructuras de Datos

Las estructuras de datos que maneja el protocolo E-D2HCP son las mismas que las del protocolo D2HCP. La única diferencia estriba en que la tabla de bloques de direcciones libres (*Free_IP_Blocks*) utilizada en E-D2HCP almacena además una bandera (*lost flag*) que indica la existencia de una ruta al nodo en cuestión.

## 7.4 Entrada y Salida de Nodos

Funciona de forma análoga a cómo funciona en D2HCP.

## 7.5 Sincronización

La sincronización se lleva a cabo monitorizando la *tabla de encaminamiento* del protocolo de encaminamiento OLSR. La entrada o salida de un nodo se detecta cuando OLSR añade una nueva ruta a su tabla de encaminamiento o borra una de las existentes. Al detectar la entrada o la salida de un nodo, se actualiza la tabla *Free_IP_Blocks* localmente sin intercambiar ningún mensaje. Para ello se sigue la siguiente operativa:

- El responsable de recuperar las direcciones IP que un nodo que abandona la red deja disponibles es aquel que puede unir por la derecha ese bloque libre con el suyo.

  Esto no será posible cuando el bloque que se debe recoger contiene la dirección más baja de la red. En ese caso, el nodo que recoge el bloque es aquel que puede añadirlo al suyo por la izquierda.

- Al dividir las direcciones libres en dos bloques para entregar uno de ellos a un nuevo nodo que entra a la red, el nodo que hace de servidor entrega al cliente el sub-bloque que no contiene su propia dirección IP.

Al detectar el ingreso de un nuevo nodo se crea una nueva entrada en la tabla para él y se actualiza el bloque de direcciones libres del nodo que le facilitó su dirección IP. Para saber quién fue ese nodo que actuó como servidor basta con buscar qué nodo tiene la dirección IP del nuevo nodo en su bloque de direcciones libres.

Cuando se detecta la salida de un nodo se debe eliminar su entrada y actualizar la del nodo a quien corresponden las direcciones IP que han quedado disponibles.

Hasta aquí el procedimiento es como en D2HCP. Ahora bien, E-D2HCP utiliza en el proceso de sincronización la bandera *lost flag* de la tabla *Free_IP_Blocks*.

Cuando OLSR pierde un nodo A, E-D2HCP examina su tabla *Free_IP_Blocks*.

Si la bandera de pérdida para esta entrada es verdadera la entrada se borra. En este caso el bloque de direcciones asignado al nodo A es recuperado por el nodo que pueda recuperar las direcciones por la derecha. Si no hay nodo que pueda recogerlas por la derecha, el nodo que puede acomodar las direcciones por la izquierda será el encargado de recoger las direcciones libres.

Si el valor de la bandera de pérdida es falso, se actualiza su valor a verdadero, no eliminándose la citada entrada.

Esta técnica previene fallos de OLSR y cambios en la ruta.

## 7.6 Inicialización

La inicialización de la red puede comenzar con un solo nodo o con un grupo de nodos. Si un nodo cliente no encuentra ningún vecino elige un rango de direcciones y genera un identificador de red (*NetworkId*). El nodo configura la interfaz de red con esta información. Si hay más de uno, todos cambian al estado *Synchronizing*. Posteriormente, ese grupo de nodos selecciona un subgrupo, los denominados precursores de la red. Estos precursores eligen un rango de direcciones y un identificador de red. A partir de este momento los nodos precursores distribuyen las direcciones a los otros nodos.

Cuando un nodo quiere ingresar a la red (nodo cliente) envía un *Server_Discovery* (en *broadcast*) y pasa al estado *SearchingServer* a la espera de mensajes *Server_Offer*.

El nodo cliente permanece en este estado durante un tiempo *Search_Serv_Time*. Una vez pasado este intervalo, el nodo cliente verifica si ha recibido algún mensaje *Server_Offer*.

Si no ha recibido ningún mensaje, incrementa el contador *nRetr* y espera nuevamente un intervalo de tiempo *Search_Serv_Time* esperando recibir mensajes *Server_Offer*.

Este proceso se realiza hasta que *nRetr* llega al valor *Max_Disc_Retries*. Si el contador *nRetr* alcanza el valor *Max_Disc_Retries* y no recibe ningún mensaje *Server_Offer*, se considera que no existe ninguna red en el entorno y el nodo cliente inicia una nueva red.

Sólo si recibe respuesta de nodos en estado *SearchingServer*, esto es, de nodos que no han encontrado ningún nodo que haya comenzado o finalizado el proceso de configuración le envía un mensaje *Client_Synchronization* pasando al estado *Synchronizing*. En este estado deciden cuál de todos ellos tiene prioridad en el proceso de inicialización de la red teniendo en cuenta la antiguedad.

## 7.7 Fusión de Redes

Durante el funcionamiento normal de una red ad hoc algunos nodos pueden separarse de la red formando otra red o unirse a una diferente. En estos procesos el identificador de red (*NetworkId*) desempeña un importante papel. Este identificador se compone de los siguientes campos:

- **IP_Originator_Node**: Dirección MAC del nodo que genera la red.
- **IP_Originator_Node**: Dirección IP del nodo que genera la red.
- **Random Number**: Número aleatorio.

El identificador de red se ha elegido así ya que la probabilidad de que dos nodos tengan la misma dirección MAC es baja. Además, la probabilidad de que dos nodos que tengan la misma dirección MAC tengan la misma dirección IP es todavía menor.

El *NetworkId* de una red cambia cada vez que el nodo que tiene la dirección MAC y la dirección IP que conforman el *NetworkId* se va de la red.

Al cambiar el *NetworkId* se asegura que las redes tengan *NetworkId* únicos.

Para detectar la fusión de la red, los nodos transmiten periódicamente mensajes *Hello* a sus vecinos. Estos mensajes *Hello* contienen la dirección IP del nodo y el *NetworkId* de la red a la que pertenece el nodo emisor.

Cada vez que un nodo X recibe un mensaje *Hello* de un nodo vecino Y que contiene un *NetworkId* diferente al suyo, detecta una fusión de redes y responde al nodo Y con el mensaje *Merge_Network* adjuntando su tabla *Free_IP_Blocks*.

Cuando el nodo Y recibe el mensaje *Merge_Network* del nodo X, también detecta la fusión de la red.

Si la red del nodo X tiene más direcciones IP que la del nodo Y, entonces el nodo Y se convierte en *MergeAgent* y el nodo X se denomina *Co-MergeAgent*.

El nodo *MergeAgent* es el responsable de la reconfiguración de la red resultante de la fusión en términos de redistribución de direcciones IP libres y de la eliminación de las direcciones IP duplicadas.

El nodo *MergeAgent* realiza una sincronización con OLSR para actualizar su tabla *Free_IP_blocks* y obtener la información actualizada de la configuración de su red, es decir, las direcciones IP asignadas a los nodos y sus correspondientes bloques *Free_IP_Blocks*.

Una vez que el *MergeAgent* tiene la información de configuración de su red y de la red del nodo *Co-MergeAgent*, detecta todas las direcciones IP conflictivas y el número de nodos con direcciones IP duplicadas entre los nodos de la red resultante de la fusión.

El nodo *MergeAgent* invalida la dirección IP de los nodos de su red enviándoles el mensaje *IP_Address_Invalidate*.

Los nodos que reciben el mensaje *IP_Address_Invalidate* reinician el proceso de auto-configuración en la red con la que se va a realizar la fusión.

Al enviar el mensaje *Server_Discovery* para iniciar la configuración de la dirección IP se debe colocar el campo *N* a 1 y el campo *NetworkId* con el identificador de la red a la que se quiere unir.

## 7.8 Formato de los Mensajes

La figura 7.1 muestra la cabecera de los mensajes enviados por el protocolo E-D2HCP.

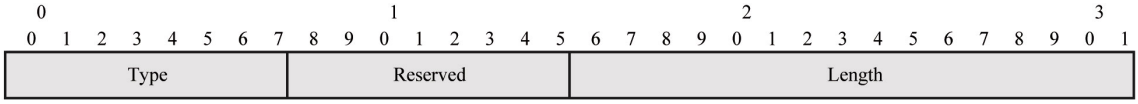



Figura 7.1: Cabecera del paquete de E-D2HCP

Los campos son los siguientes:

- **Type**: Tipo de mensaje. Puede presentar los siguientes valores:
  - 1: *Server_Discovery*
  - 2: *Server_Offer*
  - 3: *Server_Poll*
  - 4: *IP_Assigned*
  - 5: *IP_Range_Request*
  - 6: *IP_Range_Return*
  - 7: *Hello*
  - 8: *Merge_Network*
  - 9: *IP_Address_Invalidate*
- **Reserved**: Bits reservados para uso futuro. Se fijan a 0.
- **Length**: Longitud del mensaje.

### 7.8.1 Server_Discovery

Este mensaje es enviado por un nodo que desea configurar una interfaz de red.
La Figura 7.2 muestra el formato de este mensaje.

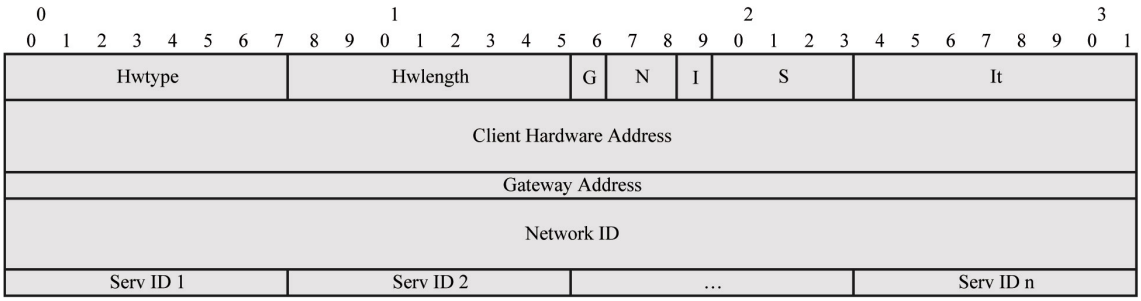



Figura 7.2: Mensaje Server_Discovery

Los campos son los siguientes:

- **Hwtype**: Tipo de hardware.
- **Hwlength**: Longitud de la dirección hardware en bytes.
- **G**: Bandera que indica si el nodo cliente dispone de una conexión a otra red.
- **N**: Campo que indica el tipo de solicitud.
  - El valor 0 indica que no existe ninguna restricción para las redes.
  - El valor 1 indica que se está buscando una red con un determinado grupo de identificadores.
  - El valor 2 indica que se busca una red creada por un dispositivo con una determinada dirección de hardware.
- **I**: Si contiene el valor 1 indica que se busca una red con acceso a Internet.
- **S**: Número de servicios requeridos por la red.
- **It**: Iteración actual del proceso de descubrimiento de servidor.
- **Client Hardware Address**: Dirección hardware de la interfaz de red que se desea configurar.
- **Gateway Address**: Campo opcional. Este campo se tiene en cuenta si la bandera G está activada.
- **Network Id**: Identificador de red cuando el valor del campo N es 1 o 2.
- **Serv Id i**: Campo opcional. Cada campo contiene el identificador de un servicio. Aparecerán tantos como indique el campo S.

### 7.8.2 Server_Offer

Este mensaje se envía en respuesta a un mensaje *Server_Discovery*. Este mensaje puede ser enviado tanto por nodos que han terminado el proceso de configuración, como por nodos que aún no han completado el proceso. Se notifica al nodo cliente el estado del nodo y, en caso de estar configurado, información relativa a la red. La Figura 7.3 muestra el formato de este mensaje.

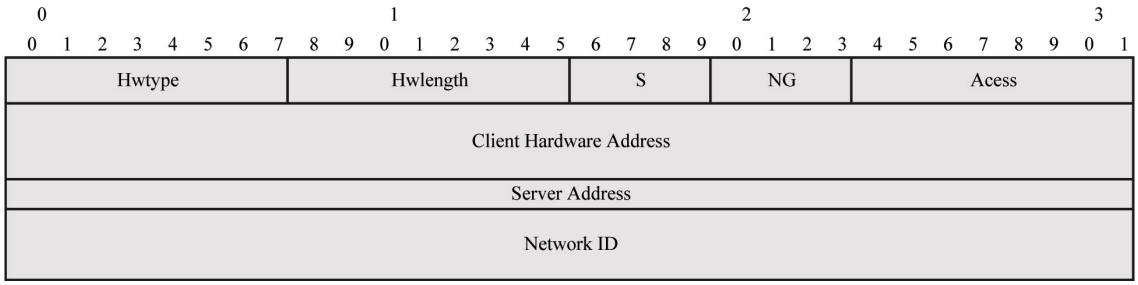



Figura 7.3: Mensaje Server_Offer

Los campos son los siguientes:

- **Hwtype**: Tipo de hardware.
- **Hwlength**: Longitud de la dirección hardware en bytes.
- **S**: Indica el estado del nodo que ha contestado el mensaje *Server_Discovery*.
- **NG**: Este campo contiene el número de *gateways* que posee dicha red.
- **Access**: Indica las restricciones de acceso a la red.
- **Client Hardware Address**: Dirección hardware de la interfaz de red que desea configurar el nodo cliente.
- **Server Address**: Dirección IP del servidor.
- **Network Id**: Identificador de red.

### 7.8.3 Server_Poll

Una vez que un nodo cliente ha recibido mensajes *Server_Offer* de uno o más nodos, envía a uno de ellos un mensaje *Server_Poll* indicándole al receptor que ha sido seleccionado como servidor. La Figura 7.4 muestra el formato de este mensaje.

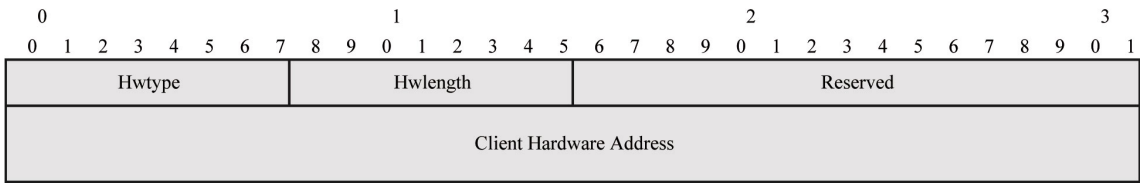



Figura 7.4: Mensaje Server_Poll

Los campos son los siguientes:

- **Hwtype**: Tipo de hardware.
- **Hwlength**: Longitud de la dirección hardware en bytes.

- **Reserved**: Bits reservados para uso futuro. Se fijan a 0.
- **Client Hardware Address**: Dirección hardware de la interfaz de red que desea configurar el nodo cliente.

### 7.8.4 IP_Assigned

Mensaje enviado por el nodo servidor seleccionado al nodo cliente. Concediéndole la administración de un rango de direcciones proporcionándole una dirección IP. La Figura 7.5 muestra el formato de este mensaje.

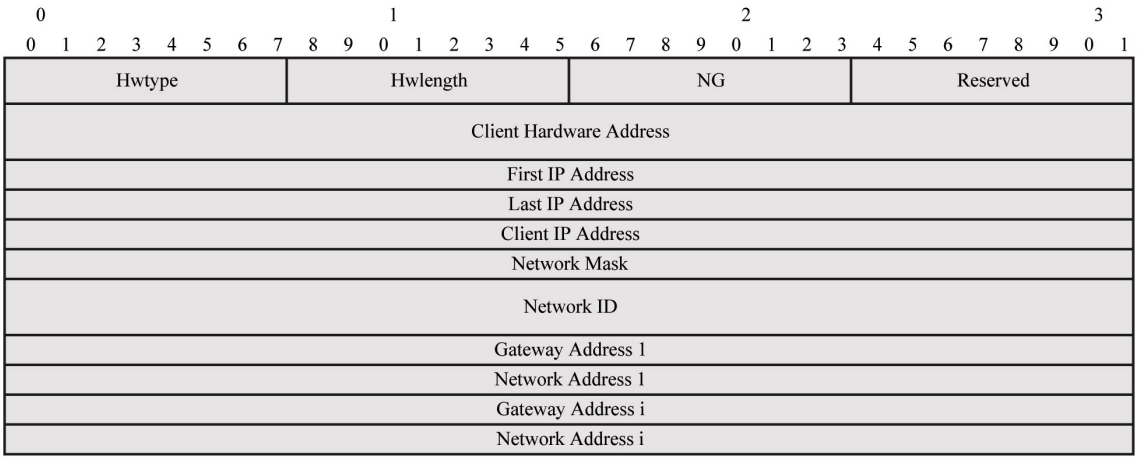



Figura 7.5: Mensaje IP_Assigned

Los campos son los siguientes:

- **Hwtype**: Tipo de hardware.
- **Hwlength**: Longitud de la dirección hardware en bytes.
- **NG**: Número de *gateways* disponibles en la red.
- **Reserved**: Bits reservados para uso futuro. Se fijan a 0.
- **Client Hardware Address**: Dirección hardware de la interfaz de red que desea configurar el nodo cliente.
- **First IP Address**: Primera dirección del rango de direcciones concedido.
- **Last IP Address**: Última dirección del rango de direcciones concedido.
- **Client IP Address**: Dirección asignada a la interfaz de red.
- **Network Mask**: Máscara de red.
- **Network ID**: Identificador de red.
- **Gateway Address i**: Dirección del *gateway* número i.
- **Network Address i**: Dirección de la red a la que da acceso el Gateway i.

### 7.8.5 IP_Range_Request

Mensaje enviado por el nodo servidor cuando las direcciones ofrecidas no son propias, sino de un tercer nodo de la red. Al ser una comunicación entre dos nodos ya configurados correctamente se realiza en la capa IP. La Figura 7.6 muestra el formato de este mensaje.

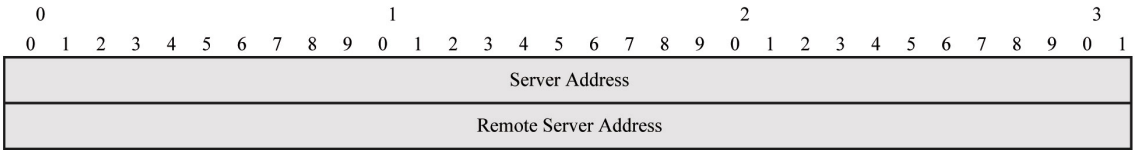



Figura 7.6: Mensaje IP_Range_Request

Los campos son los siguientes:

- **Server Address**: Dirección del nodo seleccionado como servidor. Es el nodo que inicia la solicitud de un rango de direcciones a un servidor remoto.
- **Remote Server Address**: Dirección del servidor remoto al que se solicita un rango de direcciones.

### 7.8.6 IP_Range_Return

Mensaje enviado pro el nodo servidor remoto al nodo servidor con el rango de direcciones concedido. La Figura 7.7 muestra el formato de este mensaje.

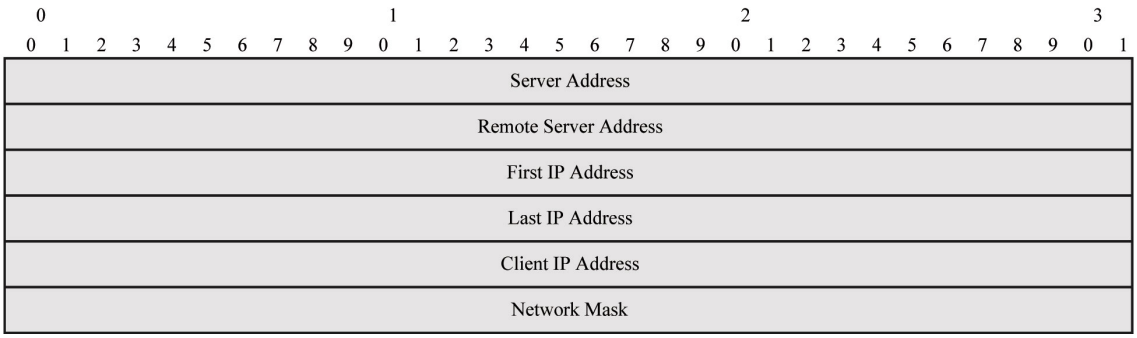



Figura 7.7: Mensaje IP_Range_Return

Los campos son los siguientes:

- **Server Address**: Dirección del nodo seleccionado como servidor. Es el nodo que inicia la solicitud de un rango de direcciones a un servidor remoto.
- **Remote Server Address**: Dirección del servidor remoto al que se solicita un rango de direcciones.
- **First IP Address**: Primera dirección del rango de direcciones.
- **Last IP Address**: Última dirección del rango de direcciones.

- **Client IP Address**: Dirección IP concedida al interfaz de red.
- **Network Mask**: Máscara de red.

### 7.8.7 Client_Synchronization

Mensaje intercambiado por nodos adyacentes para establecer cuál de ellos tendría prioridad en la inicialización de la red.La Figura 7.8 muestra el formato de este mensaje.

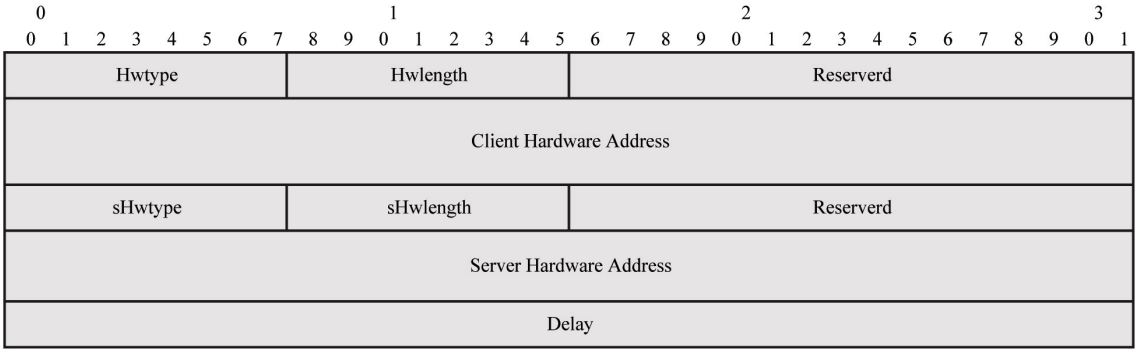



Figura 7.8: Mensaje Client_Synchronization

Los campos son los siguientes:

- **Hwtype**: Tipo de hardware.
- **Hwlength**: Longitud de la dirección hardware en bytes.
- **Reserved**: Bits reservados para uso futuro. Se fijan a 0.
- **Client Hardware Address**: Dirección hardware de la interfaz de red que desea configurar el nodo cliente.
- **Shwtype**: Tipo de hardware.
- **Shwlength**: Longitud de la dirección hardware en bytes.
- **Reserved**: Bits reservados para uso futuro. Se fijan a 0.
- **Server Hardware Address**: Dirección hardware de la interfaz de red del nodo con el que se desea realizar la sincronización.
- **Delay**: Indica el tiempo que ha pasado desde que el nodo emisor inició el proceso de configuración hasta que recibió el mensaje *Server_Offer* del otro nodo.

### 7.8.8 Hello

Mensaje enviado periódicamente para detectar posibles fusiones de redes. La Figura 7.9 muestra el formato de este mensaje.

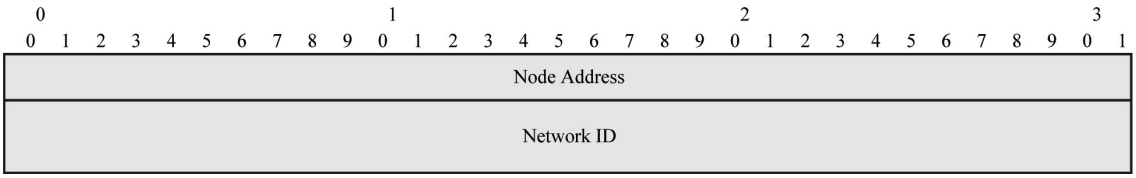



Figura 7.9: Mensaje Hello

Los campos son los siguientes:

- **Node Address**: Dirección IP del nodo emisor.
- **Network Id**: Identificador de red.

### 7.8.9 Merge_Network

Mensaje *unicast* enviado por el *Co-MergeAgent* al *MergeAgent* al detectar una fusión. La Figura 7.10 muestra el formato de este mensaje.

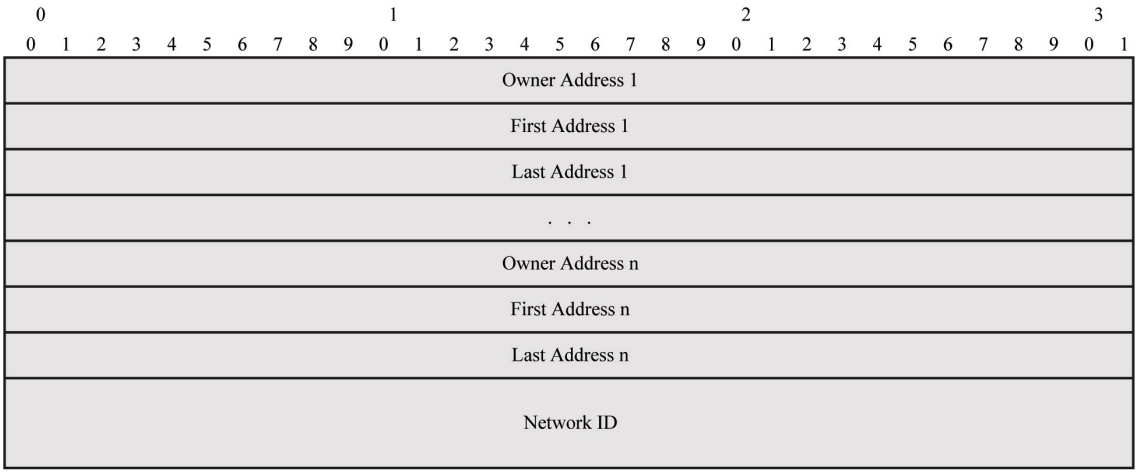



Figura 7.10: Mensaje Merge_Network

Los campos son los siguientes:

- **Owner Address 1**: Dirección IP del nodo 1 de la red.
- **First Address 1**: Dirección IP inicial del bloque de direcciones libres del nodo 1.
- **Last Address 1**: Dirección IP final del bloque de direcciones libres del nodo 1.
- La tripleta anterior (*Owner Address i*, *First Address i*, *Last Address i*) se repite para todos los nodos de la red.
- **Network ID**: Identificador de red.

### 7.8.10 IP_Address_Invalidate

Mensaje enviado por el *MergeAgent* a todos los nodos de su red para invalidar sus direcciones IP e indicarles que deben iniciar el proceso de configuración de sus direcciones IP en la red con la que se van a fusionar. La Figura 7.11 muestra el formato de este mensaje.

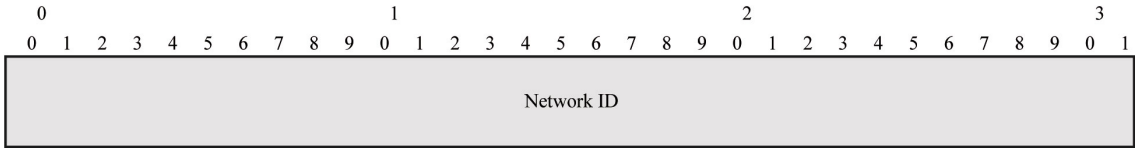



Figura 7.11: Mensaje IP_Address_Invalidate

El campo es el siguiente:

- **Network ID**: Identificador de red.

## 7.9 Temporizadores

A continuación se presentan una serie de temporizadores usados para resolver situaciones muy comunes en las redes móviles ad hoc como pérdidas de mensajes o retrasos en la llegada de los mismos. Estos temporizadores son:

- *Search_Serv_Timer*
- *Addr_All_Timer*
- *IP_Range_Request_Timer*
- *Accepted_Offer_Timer*
- *Node_Down_Timer*
- *Init_Table_Timer*
- *Init_Assign_Timer*
- *Hello_Timer*

### 7.9.1 Search_Serv_Timer

Este temporizador se inicia una vez que el nodo cliente envía el mensaje *Server_Discovery* y pasa al estado *Searching_Server* para esperar mensajes *Server_Offer* de los posibles nodos cercanos pertenecientes a una red.

### 7.9.2 Addr_All_Timer

Este temporizador se inicia una vez que el nodo cliente ha seleccionado un nodo como servidor enviándole el mensaje *Server_Poll* y pasa al estado *WaitingAllocation* esperando recibir el mensaje *IP_Assigned* el tiempo que determine este temporizador. Si este mensaje no llega, se retransmite el *Server_Poll* hasta un número determinado de intentos.

### 7.9.3 IP_Range_Request_Timer

Este temporizador se inicia cuando el nodo servidor envía un mensaje *IP_Range_Request* a otro nodo de la red.

### 7.9.4 Accepted_Offer_Timer

Este temporizador se activa en el momento de enviar un mensaje *IP_Assigned* o un mensaje *IP_Range_Return*. Durante este tiempo el nodo servidor no puede responder a solicitudes de tipo *Server_Poll* o *IP_Range_Request*. Esta restricción se levanta al expirar el temporizador (la oferta caducó sin ser aceptada) o al detectar que un nodo con la primera dirección IP de las ofrecidas ha entrado en la red (el bloque de direcciones ofrecido fue aceptado). Conviene señalar que aunque no pueda asignar direcciones IP, el nodo servidor sigue respondiendo a peticiones *Server_Discovery* con el mensaje *Server_Offer* (con R = 0). De este modo, se informa al nodo cliente de la existencia del servidor, aunque no sea capaz de asignar direcciones IP inmediatamente.

### 7.9.5 Node_Down_Timer

Este temporizador se inicia cuando OLSR borra la ruta hacia un nodo, no se elimina inmediatamente de la tabla *Free_IP_Blocks*. Si antes de que el temporizador expire se vuelve a descubrir una ruta hacia el nodo, eso quiere decir que desapareció momentáneamente, pero no abandonó la red no eliminándose de la tabla *Free_IP_Blocks*. En el caso de que el temporizador expire y no se haya recuperado una ruta, se da al nodo por perdido y se elimina su entrada de la tabla *Free_IP_Blocks*, actualizando las que correspondan.

### 7.9.6 Init_Table_Timer

El nodo cliente activa este temporizador al recibir la tabla *Free_IP_Blocks* en el mensaje *IP_Assigned*. Durante este tiempo la tabla contiene nodos para los que OLSR aún no tiene una ruta conocida. Al expirar el temporizador se comprueba qué nodos de la tabla *Free_IP_Blocks* no tienen entrada en la tabla de rutas de OLSR: esos nodos se eliminan (actualizando las entradas que correspondan), ya que son nodos que pertenecían a la red al recibir la tabla y la han abandonado antes de que OLSR supiese de su existencia.

### 7.9.7 Init_Assign_Timer

Este temporizador lo usa tanto el nodo cliente como el servidor cuando han recibido o asignado, respectivamente, un bloque de direcciones IP. Es decir, el servidor lo inicializa al comprobar la entrada de un nodo con la primera dirección IP del bloque ofrecido en el mensaje *IP_Assigned* y el cliente lo inicia tras recibir el mensaje *IP_Assigned* y configurar su dirección. Así se da tiempo a que toda la red pueda actualizar su tabla *Free_IP_Blocks* antes de que se produzcan más cambios. Durante este tiempo ignoran los mensajes *Server_Poll* o *IP_Range_Request*, aunque se responden a los *Server_Discovery*.

### 7.9.8 Node_Down_Assign_Timer

Este temporizador se pone en marcha cuando un nodo ya configurado detecta la salida de otro y comprueba que le corresponde recoger las direcciones IP que quedan libres. Más concretamente, el temporizador se activa cuando se detecta la eliminación de este nodo en la tabla de encaminamiento OLSR, es decir, se activa al mismo tiempo que

el temporizador *Node_Down_Timer*. Hasta que no expire el nodo ignora las peticiones *Server_Poll* e *IP_Range_Request*. De este modo se da un margen de tiempo para asegurarse de que todos los nodos en la red detectan la salida mencionada y actualizan su tabla *Free_IP_Blocks*. Consecuentemente, la duración de este temporizador debe ser mayor que la del *Node_Down_Timer* para asegurar que el resto de nodos de la red no sólo han detectado la eliminación de una ruta sino que también la han eliminado de la tabla *Free_IP_Blocks*. El nodo sigue respondiendo a los mensajes *Server_Discovery* con el mensaje *Server_Offer* (R = 0).

### 7.9.9 Hello_Timer

Este temporizador es utilizado en el proceso de fusión de redes. Se inicia después de enviarse el mensaje *Hello*. El valor del temporizador corresponde al intervalo de tiempo entre dos mensajes *Hello* consecutivos.

## 7.10 Parámetros de Ajuste

A continuación se presentan una serie de parámetros cuyos valores se fijan en función de las métricas que queremos optimizar en el protocolo. Al igual que los temporizadores se utilizan para resolver situaciones muy comunes en las redes móviles ad hoc como pérdidas de mensajes o retrasos en la llegada de los mismos. Estos parámetros son:

- *Max_Disc_Retries*
- *Max_All_Retries*
- *Rrequest_Max_Retries*

### 7.10.1 Max_Disc_Retries

Indica el número de intentos del que dispone el nodo cliente para reenviar mensajes *Server_Discovery* si no ha recibido ningún mensaje *Server_Offer*, ya sea por pérdidas de mensajes o porque no hay ningún nodo servidor. Si pasado este número de intentos el nodo cliente sigue sin recibir mensajes, inicia su propia red.

### 7.10.2 Max_All_Retries

Indica el número máximo de intentos que tiene el nodo cliente para reenviar el mensaje *Server_Poll*. Si se supera este número, revisará los mensajes *Server_Offer* recibidos y seleccionará otro nodo como servidor enviándole el mensaje *Server_Poll*.

### 7.10.3 Rrequest_Max_Retries

Indica el número máximo de intentos que tiene el nodo servidor para reenviar el mensaje *IP_Range_Request* al expirar el temporizador *IP_Range_Request_Timer*.

## 7.11 Diagramas de Estado

En los siguientes apartados se muestran y explican los diagramas de estados que rigen el comportamiento de ambos tipos de nodos.

### 7.11.1 Nodo Servidor

Se denomina nodo servidor a todo los nodo de la red correctamente configurado, es decir, que posee una dirección IP válida con la que se comunica con el resto de nodos y un bloque de direcciones IP libres. Con este bloque de direcciones libres facilita el acceso a nuevos nodos.

El diagrama de estados se muestra en la Figura 7.12.

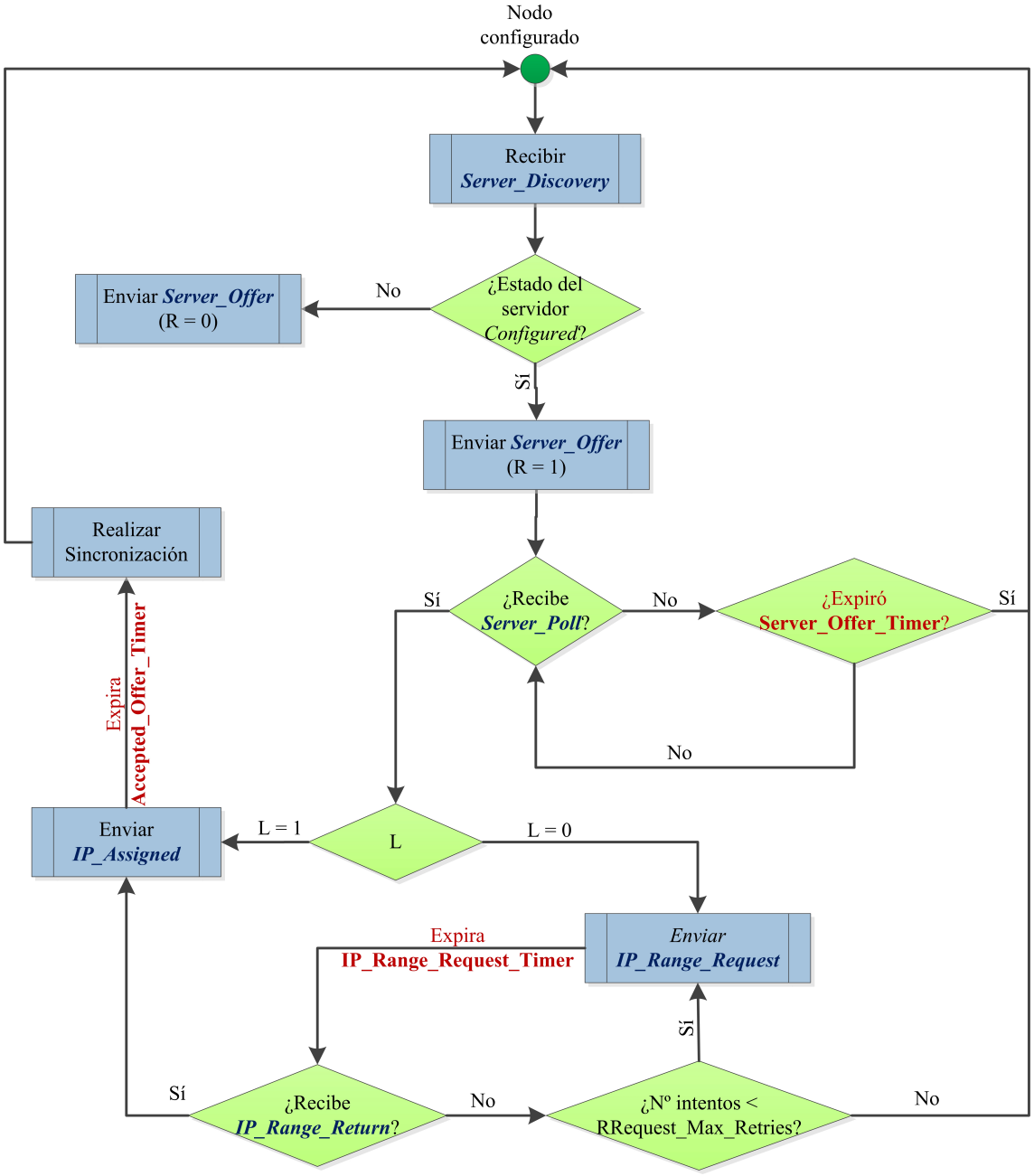



Figura 7.12: Diagrama de estados del nodo servidor

Un nodo servidor está a la espera de mensajes *Server_Discovery*. Responde siempre con un mensaje *Server_Offer*, pero dependiendo de si el estado actual del nodo es de tipo

*ready* o *not_ready*, responde dando al campo R (*Ready*) el valor 1 ó 0. De este modo, un nodo siempre anuncia su presencia, aunque en ese momento no sea capaz de asignar direcciones IP a un cliente.

El nodo servidor espera un tiempo determinado por el temporizador *Server_Offer_Timer* la recepción de un mensaje *Server_Poll* para confirmar que el nodo cliente a quien envió el *Server_Offer* le ha elegido como su servidor para el proceso de autoconfiguración.

Si el temporizador *Server_Offer_Timer* expira y el nodo servidor no ha recibido el mensaje *Server_Poll* vuelve a su estado inicial esperando mensajes *Server_Discovery*.

Tras la recepción del mensaje *Server_Poll*, realiza una de las siguientes acciones:

- Si el nodo servidor tiene las direcciones IP disponibles localmente (L = 1 en el mensaje *Server_Offer*) envía un mensaje *IP_Assigned* al nodo cliente.
- Si las direcciones ofrecidas por el nodo servidor no eran propias (L = 0 en el mensaje *Server_Offer*) debe pedirlas al nodo de la red que las contiene con el mensaje *IP_Range_Request* y queda a la espera de recibir un mensaje *IP_Range_Return*. Cuando reciba este mensaje (en el que un nodo de la red le indica un bloque que puede ofrecer al nodo cliente en espera) le envía un mensaje *IP_Assigned* al nodo cliente.

Cuando el nodo servidor envía un mensaje *IP_Assigned* al nodo cliente se activa el temporizador *Accepted_Offer_Timer* para detectar que el nodo el nodo cliente ha configurado correctamente.

Si aparece un nuevo nodo en la red usando la primera dirección IP del bloque ofrecido en el mensaje *IP_Assigned* o *IP_Range_Return*, el nodo cliente terminó de configurarse.

Si el nodo cliente no fue capaz de terminar el proceso de autoconfiguración, el temporizador *Accepted_Offer_Timer* expira y el nodo cliente debe reiniciar el proceso de configuración de direcciones IP.

### 7.11.2 Nodo Cliente

Hay 5 posibles estados por los que pueden pasar los nodos durante el proceso de configuración de direcciones IP:

- **SearchingServer**: Estado que indica que el nodo cliente ha enviado un mensaje *Server_Discovery*.
- **WaitingAllocation**: Estado del nodo cliente cuando ha encontrado un nodo servidor que le puede proporcionar un rango de direcciones válido.
- **Synchronizing**: Estado en el que se encuentran dos nodos adyacentes que no han encontrado ningún nodo que haya inicializado la red.
- **Suspended**: Estado del nodo cliente cuando ningún nodo de los que ha respondido al mensaje *Server_Discovery* ha completado su proceso de configuración de dirección IP.
- **Configured**: Estado del nodo cliente cuando ha finalizado satisfactoriamente su proceso de configuración de dirección IP.

En la figura 7.13 se muestran los estados por los que pasa el nodo cliente durante el proceso de autoconfiguración.

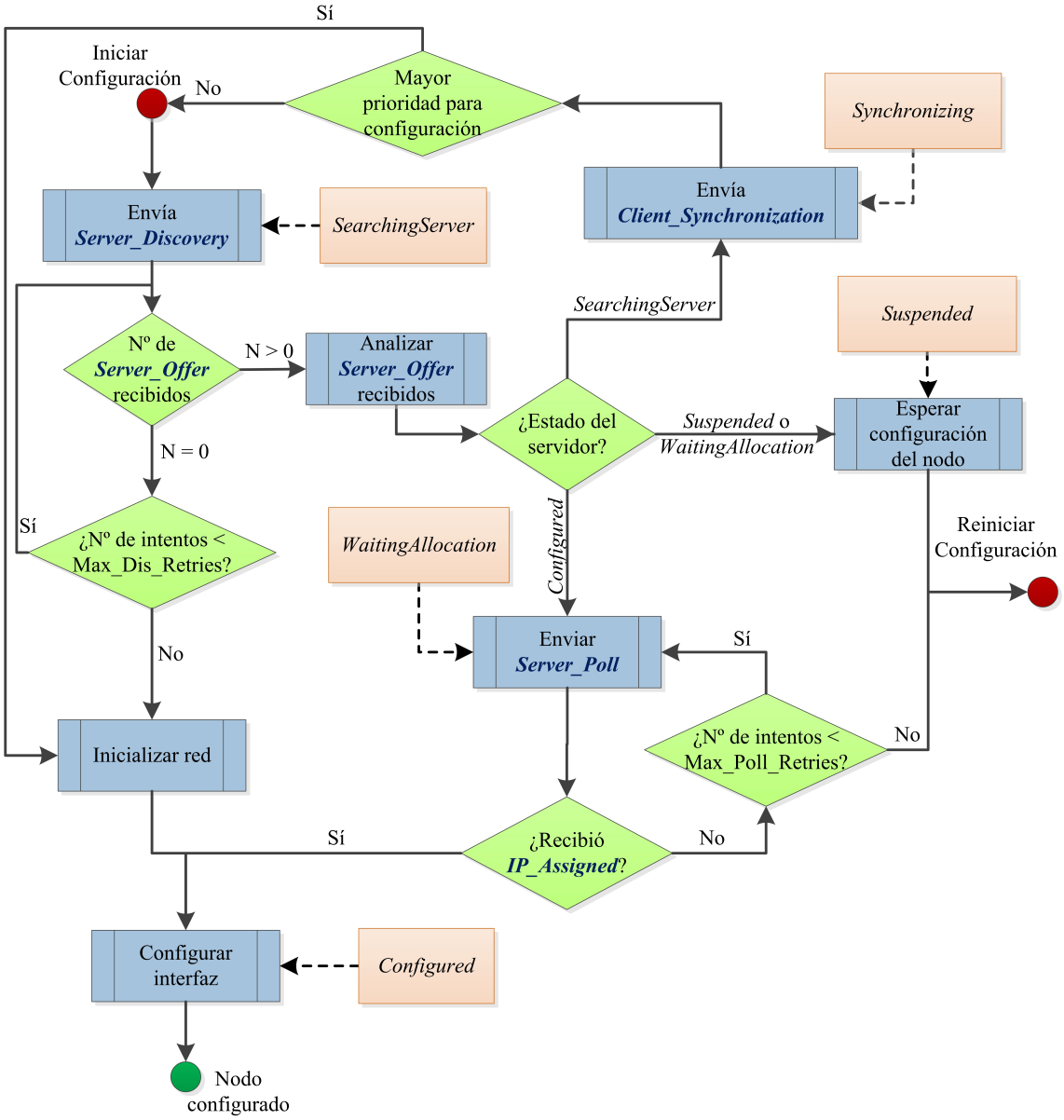



Figura 7.13: Diagrama de estados del nodo cliente

El proceso de autoconfiguración se inicia cuando un nodo desea ingresar a la red y solicita una dirección IP.

El nodo cliente envía en difusión (*broadcast*) un mensaje *Server_Discovery* y pasa al estado *SearchingServer* a la espera de mensajes *Server_Offer*. El nodo cliente permanece en este estado durante un tiempo *Search_Serv_Time*. Una vez pasado este intervalo, el nodo cliente verifica si ha recibido algún mensaje *Server_Offer*.

Si no ha recibido ningún mensaje, incrementa el contador *nRetr* y espera nuevamente un intervalo de tiempo *Search_Serv_Time* esperando recibir mensajes *Server_Offer*.

Este proceso se realiza hasta que *nRetr* llega al valor *Max_Disc_Retries*. Si el contador *nRetr* alcanza el valor *Max_Disc_Retries* se analizan las respuestas recibidas:

- Si alguna de las respuestas recibidas procede de un nodo en estado *WaitingAllocation*, dicho nodo ha encontrado un servidor y ha solicitado una dirección para su interfaz de red. Por tanto, el nodo cliente pasa al estado *Suspended* en espera de que el nodo seleccionado como servidor complete su proceso de configuración.

- Si no existe ningún nodo en estado *WaitingAllocation* pero existen nodos en estado *Suspended*, el nodo cliente pasa también al estado *Suspended*. Esto significa que o bien el nodo vecino ha encontrado un nodo que está realizando la configuración de su interfaz, o bien existe una cadena de nodos en estado *Suspended* a partir del nodo vecino que llega hasta un nodo que ya ha comenzado el proceso. Por este motivo, el nodo cliente suspende durante un período de tiempo el proceso de configuración de la dirección de la interfaz de red, para que los nodos vecinos puedan finalizar su propio proceso. Cuando el tiempo de espera finaliza, el nodo cliente reinicia el proceso pasando al estado *SearchingServer*.

- Si las respuestas proceden de nodos en estado *SearchingServer*, el remitente del mensaje no ha encontrado ningún nodo que haya comenzado o finalizado el proceso de autoconfiguración habiendo iniciado el proceso de crear una nueva red para lo cual envía un mensaje *Client_Synchronization* pasándose al estado *Synchronizing*. En este estado se decide cuál de los nodos tiene prioridad para inicializar la red.

- Si el contador *nRetr* alcanza el valor *Max_Disc_Retries* y no recibe ningún mensaje *Server_Offer*, se considera que no existe ninguna red en el entorno y el nodo cliente inicializa una nueva red.

Si el nodo cliente ha recibido mensajes *Server_Offer* se analiza el estado de los nodos que respondieron y se verifica que existan nodos en estado *Configured*:

1. Se comprueba el número de redes de las que se ha recibido respuesta verificando el *NetworkId* del nodo servidor y se selecciona una para unirse a ella.

2. De entre los nodos que responden se elige uno como nodo servidor teniendo prioridad aquel que tenga mayor número de direcciones libres propias.

Una vez seleccionado un nodo servidor se le notifica a éste su elección enviándole un mensaje *Server_Poll*.

El nodo cliente pasa al estado *WaitingAllocation* mientras espera recibir un mensaje *IP_Assigned* durante un período de tiempo *Addr_All_Time*.

Si tras este período de tiempo el nodo cliente no ha recibido ninguna respuesta se incrementa el valor del contador *nRetr* del estado *WaitingAllocation*.

Mientras este contador sea inferior a *Max_All_Retries*, se reenvía el mensaje *Server_Poll*.

En el supuesto de que se alcance el valor *Max_All_Retries* se pasa al estado *Searching-Server*.

Si el nodo seleccionado como servidor ofrece un rango de direcciones que no son propias envía un mensaje *IP_Range_Request* al nodo dueño del rango (nodo remoto) solicitando el rango de direcciones ofrecido. El nodo remoto responde al nodo servidor con el mensaje *IP_Range_Return* entregándole el rango solicitado.

El nodo servidor envía el mensaje *IP_Assigned* concediéndole la administración de un rango de direcciones al nodo cliente y proporcionándole una dirección para configurar el interfaz de red identificada por la dirección hardware. A continuación, el nodo cliente pasa al estado *Configured*.

## 7.12 Síntesis del Capítulo

El principal objetivo de este capítulo ha sido la especificación de la extensión del protocolo de autoconfiguración para redes móviles ad hoc denominada E-D2HCP. Se han enunciado las características del citado protocolo así como sus diferencias respecto a su predecesor, el protocolo D2HCP. Se ha descrito la entrada y salida de nodos, analizando la alta disponibilidad del protocolo y la posibilidad de concurrencia durante la entrada de nodos, así como la simplicidad en la salida. A continuación, se ha comentado la gestión de las estructuras de datos del protocolo y su sistema de sincronización que, al apoyarse en el protocolo de encaminamiento OLSR al igual que D2HCP, genera una sobrecarga nula en el tráfico de la red con respecto al generado por el protocolo OLSR. Asimismo, se ha prestado especial atención a la fusión de redes, aspecto éste no contemplado por D2HCP. Por último, se han analizado los mensajes y los temporizadores de E-D2HCP, así como el diagrama de estado de los nodos cliente y servidor.

# Capítulo 8

# Protocolo COD-OLSR

Este capítulo presenta una extensión de seguridad de OLSR que proporciona integridad a OLSR ante los ataques de Generación de Mensajes Incorrectos en sus dos modalidades (Suplantación de Identidad y Suplantación de Enlaces). El protocolo se ha denominado OLSR Codificado o, más concretamente, COD-OLSR, como se cita en lo sucesivo, expresión que corresponde al acrónimo de su denominación inglesa *Coded-Optimized Link State Routing*. El capítulo comienza con una breve introducción donde se pone de manifiesto la necesidad de realizar extensiones de seguridad a los protocolos de encaminamiento para redes móviles ad hoc. Posteriormente, se proporciona una completa especificación de la extensión desarrollada. El capítulo finaliza con una breve síntesis de lo expuesto en el mismo.

## 8.1 Introducción

La mayoría de los protocolos de encaminamiento para redes móviles ad hoc están diseñados sin tener en cuenta el posible comportamiento malicioso de alguno de los nodos, lo que puede ser aprovechado para vulnerar la seguridad de la red. En otras palabras, el diseño de los protocolos de encaminamiento para redes móviles ad hoc no suele contemplar, en la mayoría de los casos, entornos hostiles. Consecuentemente, es común añadir extensiones de seguridad a posteriori. Uno de los más importantes protocolos de encaminamiento es *Optimized Link State Routing* (OLSR) que, en su especificación, asume la confianza de todos los nodos en la red, por lo que es vulnerable ante diferentes clases de ataques, que pueden alterar el comportamiento del protocolo, de ahí la necesidad de una extensión segura de este protocolo. Existen diversas técnicas para proporcionar *integridad* al protocolo OLSR. Como se ha comentado en el capítulo 5, una de las más extendidas es el uso de firmas digitales para la autenticación de los mensajes de encaminamiento OLSR. Sin embargo, esta técnica introduce una sobrecarga importante.

## 8.2 Especificación

COD-OLSR tiene como objetivo detectar los ataques de Generación Incorrecta de Mensajes, haciendo que sea más difícil la suplantación de identidad. El protocolo que se especifica presenta un esquema que permite mantener la integridad de los mensajes de control de OLSR con una sobrecarga mínima. Como se ha indicado en el apartado 5.4.2, la amplia mayoría de los ataques sobre OLSR se centran en los mensajes de control *Hello* y TC, de ahí que el protocolo desarrollado se haya centrado en estos dos.

El mecanismo de COD-OLSR consta de dos fases: Codificación y Verificación. Estas dos fases están interrelacionadas, de tal forma que la integridad de un mensaje de control codificado en el emisor es comprobada por el receptor, como se puede ver en la Figura 8.1.

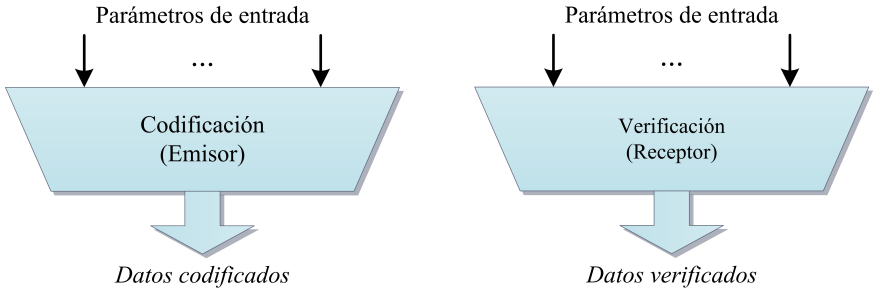



Figura 8.1: Proceso de codificación y verificación

La fase de codificación utiliza operaciones binarias (complemento a uno y XOR) que transforman la información en números de 32 bits. En este proceso de codificación se utilizan diversas de funciones que permiten generar tres campos que se añaden a la cabecera de los mensajes de control tal y como se indica en la Figura 8.2.

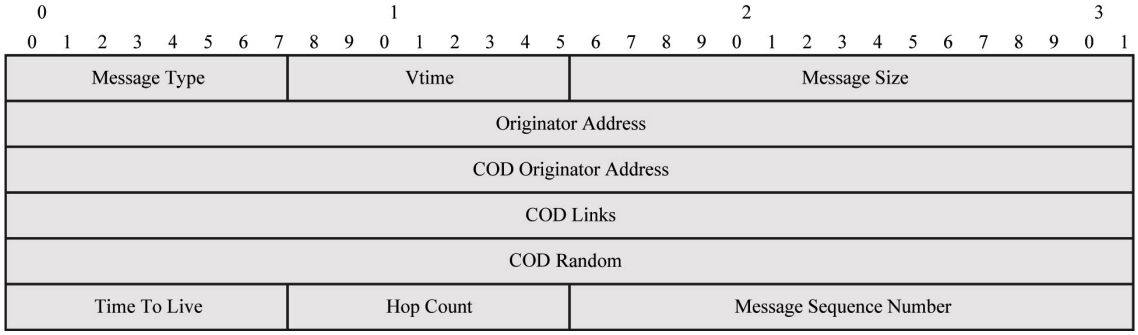



Figura 8.2: Cabecera de los mensajes COD-OLSR

Estos campos son: *COD Originator Address*, *COD Links* y *COD Random*, totalizando 12 bytes, por lo que la sobrecarga introducida es prácticamente despreciable. Cada uno de estos campos tiene una determinada funcionalidad.

Así, para prevenir la suplantación de identidad se genera *COD Originator Address* mediante la función *GenerateCodOA* según la ecuación 8.1.

$$GenerateCodOA = \overline{OA \oplus MSN \oplus \overline{RND}} \tag{8.1}$$

donde:

- **Originator Address** (*OA*): Dirección de red que el nodo tiene configurada.
- **MSN**: Número de secuencia de 2 bytes.
- **Random** (*RND*): Número aleatorio de 32 bits generado por la función *GenerateRandom*. Esta última función genera números teniendo en cuenta el tiempo actual del sistema proporcionando así mayor aleatoriedad.

La Figura 8.3 resume el proceso anterior.

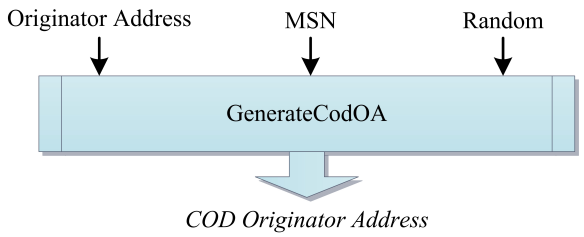



Figura 8.3: Generación de COD Originator Address

Del mismo modo, para evitar la suplantación de los enlaces se genera *COD Links* mediante la función *GenerateCodLink* según la ecuación 8.2.

$$GenerateCodLink = \overline{OA \oplus Size \oplus \overline{LINK} \oplus MSN \oplus \overline{RND}} \tag{8.2}$$

donde:

- **Originator Address** ($OA$): Dirección de red que el nodo tiene configurada.
- **Random** ($RND$): Número aleatorio de 32 bits.
- **MSN**: Número de secuencia de 2 bytes.
- **Size**: Tamaño del mensaje (16 bits).
- **LINK**: Lista de valores de 32 bits asociados al contenido de los mensajes de control. Esta lista puede estar asociada a las direcciones de los vecinos si se trata de un mensaje de control *Hello* o a las direcciones de *MPR Selector* (MS) si se trata de mensajes de control TC. Esta lista se codifica de N a 1, es decir, se transforman varios elementos en uno por medio de las operaciones binarias según la ecuación 8.3.

$$LINK = \begin{cases} link_{i-1} \oplus \overline{link_i} & si\ i\ es\ par \\ link_{i-1} \oplus link_i & si\ i\ es\ impar \end{cases} \tag{8.3}$$

La Figura 8.4 resume el proceso anterior.

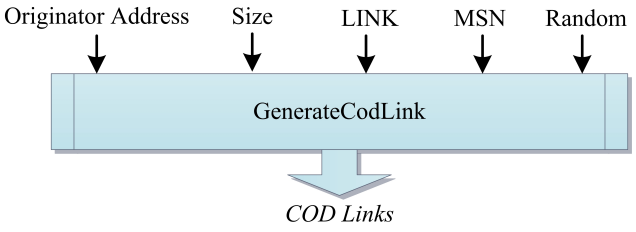



Figura 8.4: Generación de COD Links

Finalmente, para una mayor protección está *COD Random* que se crea con la función *GenerateCodRandom* según la ecuación 8.4.

$$GenerateCodRandom = \overline{OA \oplus \overline{MSN} \oplus RND \oplus LINK} \tag{8.4}$$

donde:

- **Originator Address** (*OA*): Dirección de red que el nodo tiene configurada.
- **Random** (*RND*): Número aleatorio de 32 bits sin codificar.
- **MSN**: Número de secuencia de 2 bytes.
- **LINK**: Lista de valores (direcciones IP de los nodos) que constituyen la clave de COD-OLSR. Esta lista de valores de 32 bits representa la topología actual respecto al nodo que genera el mensaje de control, que también tiene una codificación N a 1.

La Figura 8.5 resume el proceso anterior.

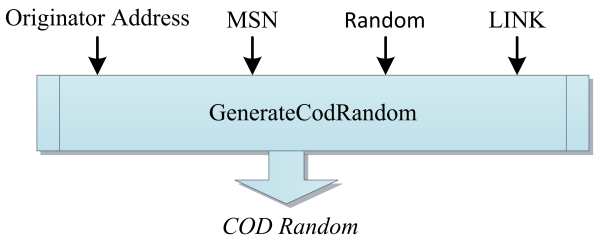



Figura 8.5: Generación de COD Random

Como se ha comentado anteriormente, la detección del ataque se realiza en el receptor (en la fase de verificación), que comprueba que los campos codificados en el emisor no se han modificado, lo que garantiza la integridad de los mensajes.

En el ejemplo de la Figura 8.6 se puede ver cómo tanto el nodo emisor como el receptor tienen en cuenta la topología.

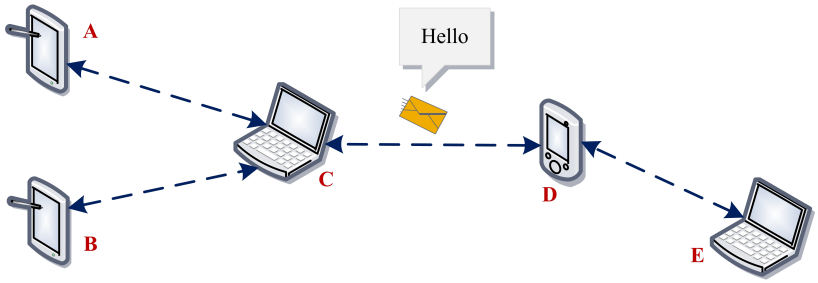



Figura 8.6: Esquema de codificación de la topología

El nodo emisor C envía un mensaje de control *Hello* al nodo D. Los vecinos a un salto de C son *LINK* = {A, B, D}. Cuando el receptor D recibe este mensaje *Hello* tiene que calcular en el proceso de verificación la topología del nodo emisor C. Para ello calcula los vecinos a dos saltos, siendo C el nodo intermedio. Para que no haya problemas de incoherencia porque la topología del nodo emisor puede cambiar después de enviar el mensaje, se ha utilizado una técnica de sincronización para que el tiempo que transcurra

sea muy pequeño (del orden de milisegundos) desde la fase de codificación del mensaje en el emisor hasta su verificación en el receptor.

De esta forma el receptor calcula qué vecinos tiene el emisor. Una vez que ha hallado la topología del nodo emisor realiza el proceso inverso: primero se calcula el número aleatorio *Random* mediante la función *RecoverCodRandom* según la ecuación 8.5.

$$RecoverCodRandom = \overline{CODRandom} \oplus OA \oplus \overline{MSN} \oplus LINK \tag{8.5}$$

La Figura 8.7 resume el proceso anterior.

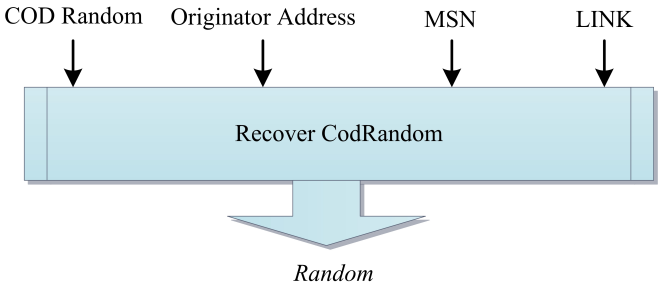



Figura 8.7: Recuperación de Random

Después, con este número aleatorio *Random* y con la información del mensaje que recibe el receptor, calcula *COD OriginatorAddress* y *COD Links*. Si los valores calculados no corresponden con los campos *COD Originator Address* y *COD Links* contenidos en el mensaje, se considera que hay un ataque y el mensaje es descartado por el receptor.

Lo más importante de este esquema es que utiliza la codificación de la topología actual del nodo emisor, por lo que el supuesto atacante, al desconocer la topología dinámica de la red, no puede interpretar el mensaje, aunque sí puede verlo. Si la topología del nodo emisor no cambia, el atacante puede aprovecharse de esta situación al monitorizar la red, pudiendo realizar la suplantación.

Conviene recordar que como medida de seguridad en la codificación de los tres campos *COD Originator Address*, *COD Links* y *COD Random* se ha utilizado el número de secuencia *MSN*.

## 8.3 Síntesis del Capítulo

El principal objetivo de este capítulo ha sido especificar una extensión del protocolo de encaminamiento OLSR, llamada COD-OLSR, que proporciona integridad al citado protocolo ante los ataques de Generación de Mensajes Incorrectos tanto en su modalidad de Suplantación de Identidad como en su modalidad de Suplantación de Enlaces, siendo una de sus principales características el hecho de que tiene en cuenta la topología actual del nodo que envía el mensaje. A diferencia de otras propuestas existentes en la literatura relativas al uso de firmas digitales para la autenticación de los mensajes de encaminamiento OLSR que introducen una sobrecarga importante, la extensión aquí definida añade una ligera sobrecarga.

# Capítulo 9

# Simulación y Resultados

Este capítulo presenta las simulaciones realizadas utilizando varios escenarios reales que comprueban la aplicabilidad de las diferentes propuestas. Para ello se ha utilizado el simulador de redes *Network Simulator 3* [NS3], uno de los más utilizados en el área. Primero se argumenta la elección del simulador de redes elegido. A continuación se describen los escenarios de simulación utilizados. Posteriormente, se comentan las métricas de simulación que se han analizado como, por ejemplo, la latencia en la asignación de direcciones IP, el número de mensajes intercambiados, la sobrecarga en el número de paquetes, la sobrecarga en el número de bytes, etc. El capítulo finaliza con una breve síntesis de lo expuesto en el mismo.

## 9.1 Elección del Simulador de Redes

Las simulaciones se utilizan como apoyo en el diseño de protocolos. Hay dos aspectos importantes que deben evaluarse antes de la realización de las mismas: uso del modelo adecuado y elección de la mejor herramienta para el modelo en cuestión.

A continuación se presentan los simuladores de redes más relevantes así como las características de cada uno de ellos:

- ***Network Simulator 2* (NS-2)**:

  *Network Simulator 2* (NS-2) [NS2] es un simulador de eventos discretos utilizado principalmente en ambientes académicos y de investigación. Las simulaciones se componen de código escrito en C++ (que es usado para modelar el comportamiento de los nodos simulados) y por secuencias de comandos *Object-Oriented Tool Command Language* (oTcl) (que controlan la simulación y especifican aspectos adicionales como la topología de la red). Este diseño fue elegido para evitar recompilaciones innecesarias cuando se hacían cambios en la estructura de la simulación ya que una frecuente recompilación del programa en C++ consumía mucho tiempo cuando salió la primera versión. Sin embargo, actualmente esto no es un problema y no es necesario sacrificar el rendimiento de la simulación para ahorrar en recompilaciones, sobre todo cuando se simula una red de gran tamaño [BHvR05].

- ***Network Simulator 3* (NS-3)**:

  Al igual que su predecesor, NS-3 [NS3] es un simulador de eventos discretos y se basa en C++ para la implementación de los modelos de la simulación. Sin embargo, NS-3 ya no utiliza secuencias de comandos oTcl para controlar la simulación evitando los problemas presentados por la combinación de C++ y oTcl en NS-2. Los escenarios

de simulación en NS-3 pueden implementarse en C++ puro y, opcionalmente, partes de la simulación se pueden realizar utilizando *Python*.

- ***Objective Modular Network Testbed in C++* (OMNeT++)**:

  En contraste con NS-2 y NS-3, OMNeT++ [OMN] no es un simulador de red por definición, sino un simulador de propósito general basado en eventos discretos. Sin embargo, se aplica sobre todo al dominio de simulación de redes, teniendo en cuenta el hecho de que su paquete *Integrated Network Enhanced Telemetry* (INET) ofrece una amplia colección de modelos de protocolos de Internet. Las simulaciones consisten en los llamados módulos simples que realizan el comportamiento de un modelo, por ejemplo, un determinado protocolo. Se pueden unir varios módulos simples para formar un módulo compuesto [BHvR05]. Al igual que NS-2 y NS-3, OMNeT++ se basa en C++ para la implementación de los módulos simples. La composición de estos módulos simples en módulos compuestos y, por tanto, la configuración de la simulación, se lleva a cabo en *Network Description* (NED), lenguaje de descripción de red de OMNeT++.

De los simuladores de red mencionados es NS-2 el más utilizado en el ámbito académico y de investigación. Sin embargo, muchos de sus usuarios se quejan de la complejidad propia del simulador y del alto consumo de recursos que lleva a la falta de escalabilidad, impidiendo la ejecución de simulaciones de redes con cientos de nodos [KÖ8]. Esto se debe a que el tiempo de simulación aumenta exponencialmente con el número de nodos de la red y además consume mucha memoria al ejecutar la simulación.

Debido a todos estos problemas se creó NS-3. Uno de sus principales objetivos fue eliminar el problema de escalabilidad y soportar la simulación de manera paralela y distribuida [HRFR06]. A pesar de que NS-3 no tiene todos lo modelos que tiene actualmente NS-2, posee más detalles de los modelos del éstandar IEEE 802.11 y es posible integrarle nuevos módulos posibilitando que el simulador se actualice, permitiéndole seguir el rápido crecimiento de las redes inalámbricas [HRFR06]. Adicionalmente, NS-3 tiene nuevas funcionalidades como son: manejo correcto de múltiples interfaces, uso de direcciones IP, genera archivos PCAP que se utilizan para el análisis, etc.

En cuanto a OMNeT++ es un simulador bien organizado, flexible y fácil de usar. Sin embargo, posee informes bastante pobres de los resultados de la simulación, por lo que los usuarios deben desarrollar el código para obtener las métricas deseadas. Tiene extensiones externas las cuales permiten proveer soporte para la simulación de redes inalámbricas. Sin embargo, sólo es posible simular algunos escenarios ya que estas extensiones no están completas, sobre todo la de movilidad, además de que la documentación todavía está en desarrollo y el análisis de las métricas de rendimiento es deficiente.

En [WvLW09] se demuestra que OMNeT++ requiere más tiempo que NS-3 para realizar una simulación, mientras que NS-2 no escala bien y, por tanto, no es adecuado para simulaciones de redes a gran escala. Asimismo, NS-3 es el simulador más eficiente con respecto al uso de memoria.

Las consideraciones anteriores determinaron la elección del simulador de redes NS-3.

## 9.2 Entorno de Simulación

Para la realización de las simulaciones de los protocolos de autoconfiguración D2HCP y E-D2HCP se ha considerado una red ad hoc aislada, es decir, una red ad hoc sin conexión a una red externa. Se considera que la red tiene direccionamiento IPv4 clase B o C y que el

rango de direcciones IP del cual se van a asignar las direcciones IP de los nodos se conoce de antemano. Como la red es aislada no es necesario configurar la dirección de la puerta de enlace predeterminada.

Para la movilidad de los nodos se ha utilizado el modelo *Random WayPoint Mobility Model* (RWP), que es el modelo más utilizado.

La Tabla 9.1 presenta los principales parámetros utilizados durantes las simulaciones. Se ha mantenido constante el número de entradas en la red por unidad de tiempo. Este factor es importante, particularmente en redes de alta densidad.

Tabla 9.1: Parámetros de las simulaciones

| **Parámetro** | **Valor** |
|---|---|
| Área de simulación | 750 x 750 $m^2$. |
| Número de nodos | De 0 a 1600. |
| Distribución de los nodos | Aleatoria. |
| Patrón de movilidad | RWP. |
| Alcance o cobertura del nodo | 125 metros. |
| Protocolo de encaminamiento | OLSR. |
| Tipo de direccionamiento | B y C. |
| Número de simulaciones | 10 |

Para las simulaciones de D2HCP se ha utilizado una red en la cual se configuran nodos entre un mínimo de 20 y un máximo de 80, según el estándar IEEE 802.11b con un rango de transmisión de 125 m y en un área de 750 x 750 $m^2$. La clase de red utilizada es de clase C de direcciones privadas con un máximo de 254 direcciones configurables. Las posiciones de los nodos que se van configurando son aleatorias. El retardo de entrada de nodos en la red es de 2 a 3 s. Los nodos se mueven en el área establecida siguiendo el modelo de movilidad RWP con velocidad de los nodos entre un mínimo de 0 y un máximo de 5 m/s, siendo el tiempo de pausa de 10 s. El protocolo de encaminamiento utilizado para la sincronización es OLSR. El tiempo de simulación es directamente proporcional al número de nodos y al retardo de entrada de los nodos en la red.

Para las simulaciones de E-D2HCP se ha utilizado una red en la cual se configuran nodos entre 50 y 250. La clase de red utilizada es de clase C de direcciones privadas con un máximo de 254 direcciones configurables. El área de simulación es de 750 x 750 $m^2$. Las posiciones de los nodos que se van configurando son aleatorias. El retardo de entrada de nodos en la red es de 2 a 3 s. La velocidad de los nodos oscila entre 0 y 5 m/s y el tiempo de pausa es de 10 s.

Para las simulaciones de COD-OLSR se ha utilizado el siguiente escenario:

- Un área de 1000 x 1000 $m^2$ y un tiempo de simulación de 30 s.
- 4 sesiones de datos (siempre los mismos pares origen/destino).
- La aplicación utilizada para la generación de los datos ha sido *Constant Bit Rate* (CBR), en la que se envían 4 paquetes de datos de 64 bytes por segundo, es decir, una tasa de envío de datos de 2048 bps.
- El modelo de movilidad utilizado ha sido RWP con una velocidad entre un mínimo de 0 y un máximo de 5 m/s, siendo el tiempo de pausa de 5 s.
- Los dispositivos de red fueron configurados según el estándar IEEE 802.11b con un rango de transmisión de 150 m.

## 9.3 Métricas de Rendimiento

Las métricas de rendimiento definidas para la evaluación de D2HCP y E-D2HCP han sido las siguientes:

- **Latencia en la Asignación de Direcciones**: Retardo que experimenta el algoritmo cuando un nodo se une a la red en el proceso de autoconfiguración o tiempo transcurrido desde que se envía el primer mensaje hasta que el nodo se configura.
- **Mensajes Intercambiados**: Número total de mensajes necesarios para el proceso de autoconfiguración.
- **Sobrecarga en el Número de Paquetes**: Número de paquetes enviados y recibidos en la red por el algoritmo de autoconfiguración.
- **Sobrecarga en el Número de Bytes**: Número total de bytes enviados y recibidos en la red por el algoritmo de autoconfiguración. La sobrecarga en el número de bytes es más representativa que la de paquetes, porque la de bytes tiene en cuenta las cabeceras de los mensajes.

Las métricas de rendimiento definidas para la evaluación de COD-OLSR han sido las siguientes:

- **Sobrecarga en el Número de Paquetes de Control**: Relación entre los paquetes de control enviados y los paquetes de datos correctamente entregados.
- **Sobrecarga en el Número de Bytes**: Relación entre el número total de bytes enviados y el número de bytes de los paquetes de datos entregados correctamente.
- **Tolerancia a Fallos del Sistema**: Esta métrica tiene en cuenta el número de mensajes de control que son descartados por COD-OLSR. La tolerancia a fallos del sistema representa la fracción de mensajes de control perdidos (no procesados por el algoritmo).
- **Ratio de Entrega de Paquetes de Datos**: Relación entre el número de paquetes de datos entregados correctamente al destino y el número total de paquetes enviados.
- **Throughput**: Volumen de información que fluye a través de un sistema. Se calcula dividiendo el total de bits entregados al destino por el tiempo de entrega de paquetes.

## 9.4 Evaluación del Protocolo D2HCP

Para evaluar las prestaciones del protocolo D2HCP, en términos de eficiencia en la asignación de direcciones IP, se han tenido en cuenta diversos factores como el impacto del incremento del número de nodos en la red y cómo éste afecta a parámetros como la latencia y la sobrecarga, considerando la frecuencia de entrada y salida de nodos en la red. Esta evaluación se ha realizado conjuntamente con la del protocolo de Mohsin & Prakash [MP02]. En las gráficas este protocolo se ha denotado simplemente con el término Prakash.

### 9.4.1 Latencia en la Asignación de Direcciones

Como se observa en la Figura 9.1, la latencia en D2HCP es inferior en todo momento a la de [MP02]. Asimismo, se observa que la latencia en D2HCP se mantiene casi constante frente a la ocupación de la red, creciendo de forma ligeramente lineal con el número de nodos, lo que permite inferir la escalabilidad del protocolo.

### 9.4.2 Número de Mensajes Intercambiados

Como se observa en la Figura 9.2, el número de mensajes intercambiados en D2HCP es inferior al requerido por [MP02]. Se observa asimismo que en un amplio rango del número de nodos el número de mensajes intercambiados es 4, lográndose por tanto direccionamiento local (la autoconfiguración se logra con la ayuda de los nodos vecinos de un salto).

### 9.4.3 Sobrecarga en el Número de Paquetes

Como se observa en la Figura 9.3, D2HCP presenta menor sobrecarga en el número de paquetes que [MP02]. Este último tiene mayor sobrecarga porque encapsula varios mensajes en un mismo paquete, hecho que no ocurre con D2HCP. Asimismo, conviene reseñar que la sobrecarga en el número de paquetes recibidos en [MP02] es mayor que en el de los enviados, porque en [MP02] se utilizan envíos en modo *broadcast*. Un mensaje que se difunde en modo *broadcast* es recibido por todos sus vecinos de un salto.

### 9.4.4 Sobrecarga en el Número de Bytes

Como se observa en la Figura 9.4, D2HCP presenta también menor sobrecarga en el número de paquetes que [MP02].

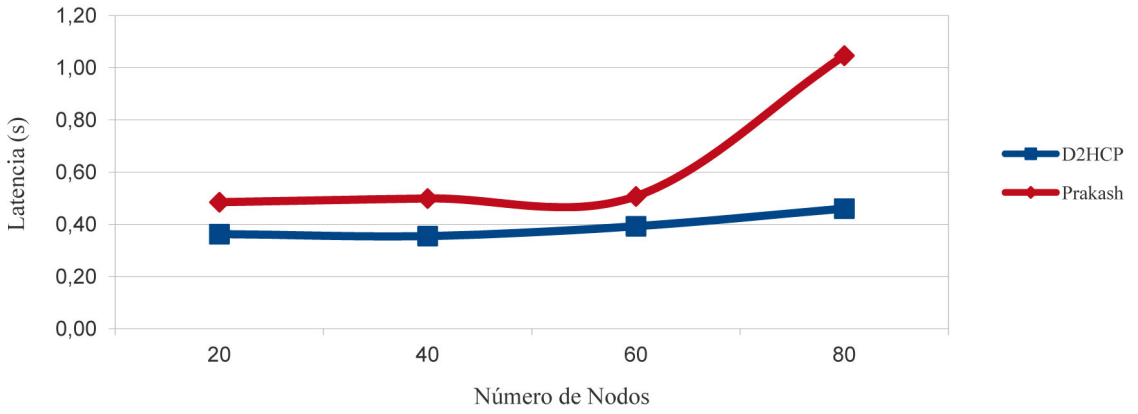



Figura 9.1: Latencia en la asignación de direcciones en una red clase C

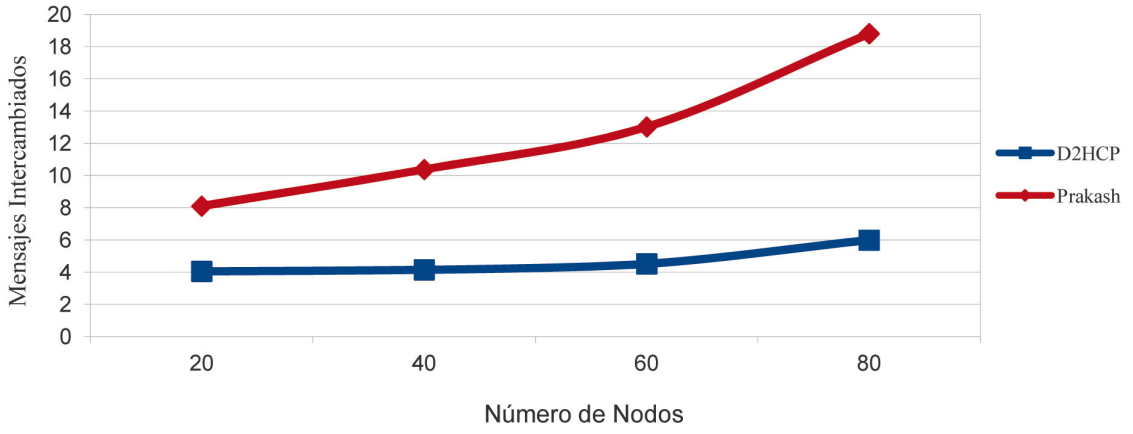



Figura 9.2: Número de mensajes intercambiados en una red clase C

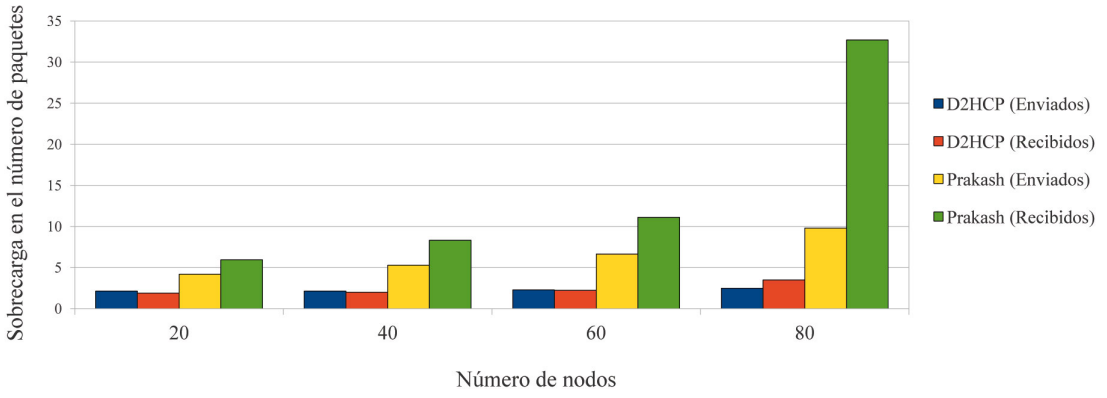



Figura 9.3: Sobrecarga en el número de paquetes en una red clase C

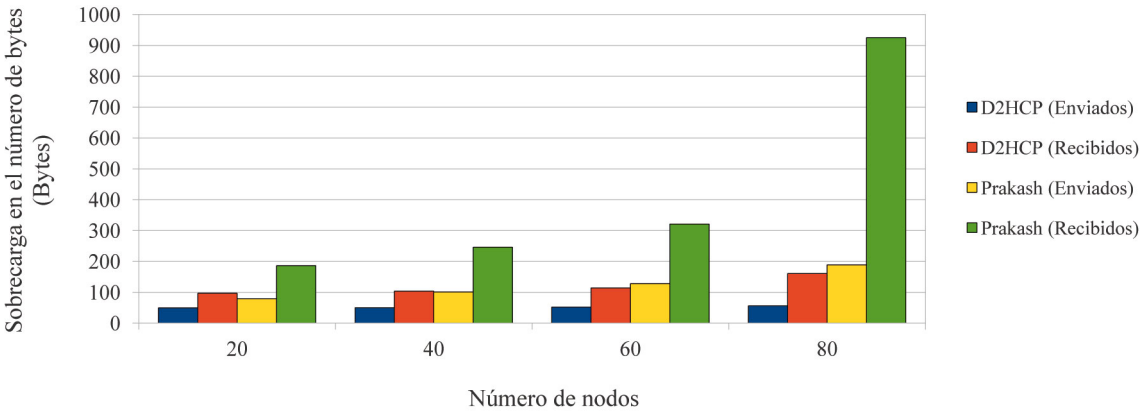



Figura 9.4: Sobrecarga en el número de bytes en una red clase C

## 9.5 Evaluación del Protocolo E-D2HCP

### 9.5.1 Latencia en la Asignación de Direcciones

Como se observa en la Figura 9.5, la latencia en la asignación de direcciones en redes clase C crece de forma más rápida que en redes clase B, especialmente a partir del 75 % de ocupación de la red. Estos resultados indican que el parámetro que condiciona la latencia es el porcentaje de direcciones ocupadas en la red. En el caso de direcciones de la clase C, al disponer de menos direcciones que en clase B, disminuye el número de asignaciones locales siendo necesario solicitar las direcciones a otro nodo, lo que aumenta el tiempo necesario para completar el proceso de asignación de dirección IP. No obstante esto, la latencia promedio en el proceso de asignación de direcciones es baja en ambas clases de direcciones.

### 9.5.2 Número de Mensajes Intercambiados

La Figura 9.6 muestra el número medio de mensajes de control emitidos en cada configuración de dirección IP. Se observa que E-D2HCP presenta una reducción en la sobrecarga de paquetes de control. En efecto, en la mayoría de los casos la configuración se realiza de forma local, es decir, un vecino asigna la dirección al nuevo nodo. En el caso de que no se puedan asignar direcciones localmente se realiza una transmisión *unicast* con el servidor elegido. El número medio de mensajes intercambiados es muy parecido en ambas clases de direcciones IPv4. A partir del 75 % de ocupación de la red en clase C hay un ligero aumento en el número medio de mensajes frente a la clase B.

### 9.5.3 Sobrecarga en el Número de Paquetes

Como se observa en la Figura 9.7, la sobrecarga en el número de paquetes es muy similar hasta el 75 % de ocupación de la red en ambas clases de direcciones. A partir del 75 % de ocupación de la red aumenta el número de paquetes recibidos sin apreciarse diferencia en las dos clases de redes analizadas.

### 9.5.4 Sobrecarga en el Número de Bytes

Como se observa en la Figura 9.8, la sobrecarga en el número de bytes se comporta de forma muy similar para ambas clases, salvo cuando el porcentaje de ocupación de la red se acerca al 100 % donde los bytes recibidos en la clase B son el doble de los recibidos en la clase C porque hay más posibilidad de direccionamiento.

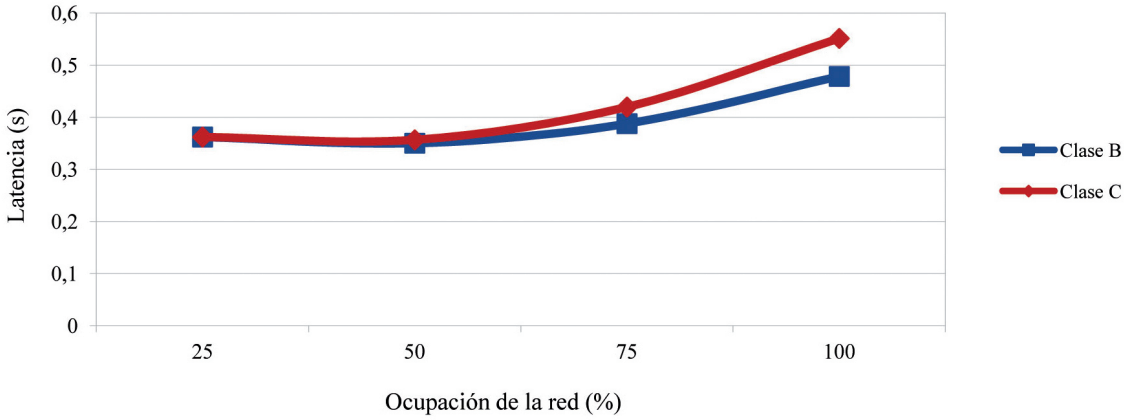



Figura 9.5: Latencia para redes clases B y C

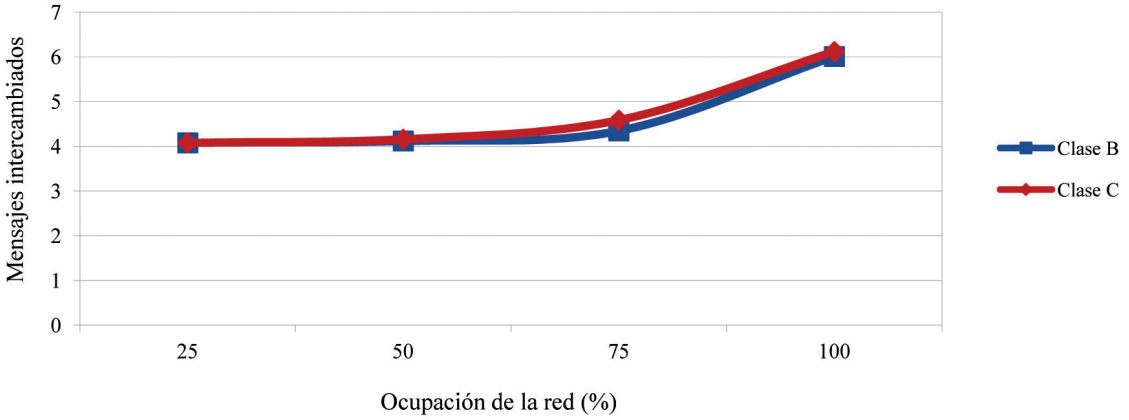



Figura 9.6: Mensajes intercambiados para redes clases B y C

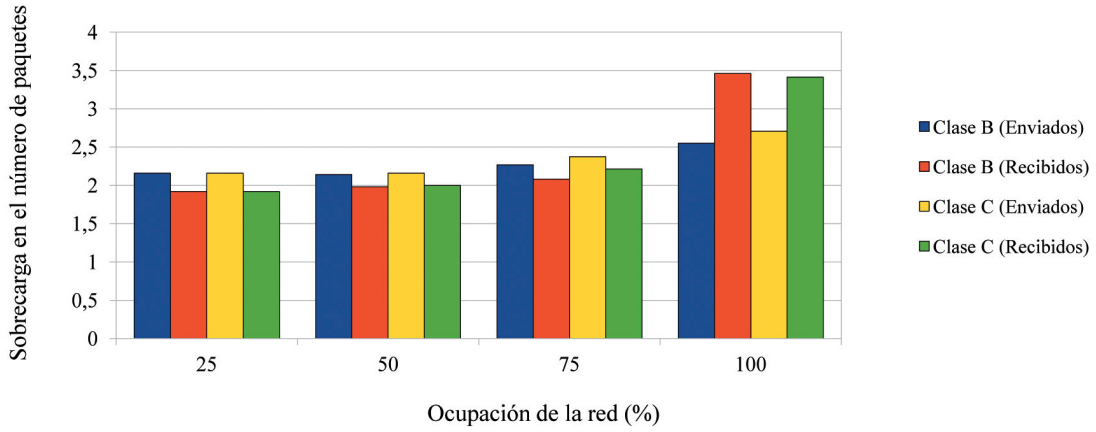



Figura 9.7: Sobrecarga en el número de paquetes para redes clases B y C

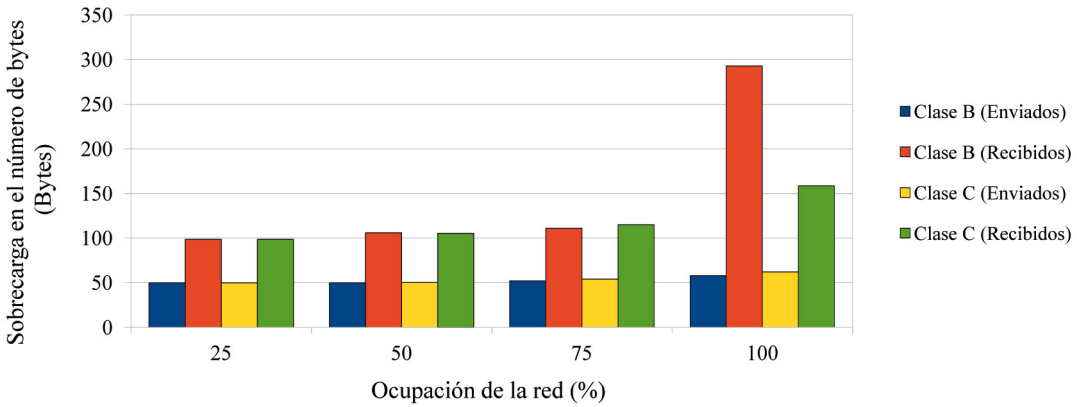



Figura 9.8: Sobrecarga en el número de bytes para redes clases B y C

## 9.6 E-D2HCP versus D2HCP

En esta sección se analiza el impacto que tiene el número de nodos en la red en las métricas de latencia y sobrecarga en los protocolos D2HCP y E-D2HCP en redes clase C.

### 9.6.1 Latencia en la Asignación de Direcciones

Como se observa en la Figura 9.9, la latencia, en promedio, disminuye más de un 50 % en E-D2HCP respecto a D2HCP. Asimismo, se observa que la latencia en E-D2HCP disminuye conforme aumenta la ocupación de la red. No obstante esto, en el caso de una red compuesta por 240 nodos (95 % de ocupación), la latencia en la asignación de direcciones IP es inferior a un segundo.

### 9.6.2 Número de Mensajes Intercambiados

Como se observa en la Figura 9.10, cuando hay pocos nodos en la red, el número de mensajes intercambiados se acerca al mínimo, ya que la configuración se realiza de forma

local (usando sólo 4 mensajes) en ambos protocolos. También se observa que E-D2HCP logra como mínimo una reducción del 25 % en la sobrecarga a partir de 150 nodos respecto a la presentada por D2HCP.

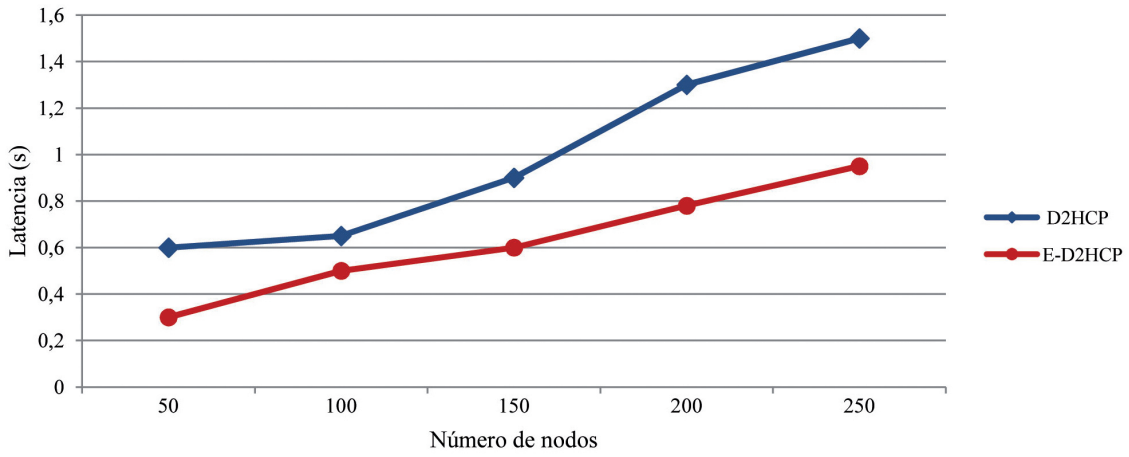



Figura 9.9: Latencia en la asignación de direcciones IPv4 en una red clase C

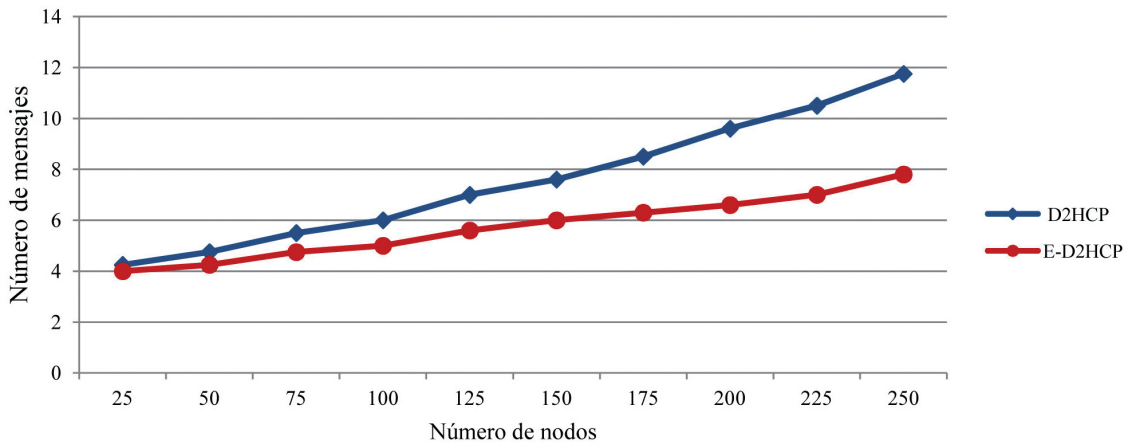



Figura 9.10: Sobrecarga en la asignación de direcciones IPv4 en una red clase C

## 9.7 D2HCP / E-D2HCP versus otros Protocolos

En este apartado se realiza una comparación de las prestaciones de D2HCP y E-D2HCP respecto a los principales protocolos de autoconfiguración existentes en la literatura. La Tabla 9.2 muestra la métricas más relevantes utilizadas tradicionalmente para esta comparativa [ZNM03].

Los protocolos presentados comparten algunas características comunes. Sin embargo, también difieren entre sí en una amplia gama de aspectos. La Tabla 9.3 presenta una comparación de las características de los protocolos de direccionamiento IPv4 agrupados

Tabla 9.2: Métricas de evaluación

| **Métrica** | **Descripción** |
|---|---|
| Garantía de la unicidad (G) | Cada nodo debe tener una dirección IP única para cada interfaz de red. |
| Sobrecarga (S) | Número de paquetes intercambiados para obtener una dirección IP. |
| Latencia (L) | Tiempo de espera de un nodo para obtener la dirección IP. |
| Dependencia del encaminamiento (D) | Los protocolos de autoconfiguración pueden funcionar de dos formas: apoyado en un protocolo de encaminamiento para permitir el encaminamiento de los nuevos nodos que ingresan a la red o de forma independiente al citado algoritmo. |
| Uniformidad (U) | Todos los nodos realizan la misma función. |

según el modo de gestión de las direcciones IP.

Como se observa la Tabla 9.3, los protocolos D2HCP y E-D2HCP presentan mejores prestaciones que el resto de protocolos de autoconfiguración con respecto a la garantía de unicidad, latencia y sobrecarga en la asignación de direcciones IP. Sin embargo, depende del protocolo de encaminamiento OLSR para la sincronización de la red. Asimismo, en los protocolos D2HCP y E-D2HCP todos los nodos pertenecientes a la red realizan las mismas funciones en la red lo que facilita la implementación de los mismos.

## 9.8 Evaluación del Protocolo COD-OLSR

En esta sección se analiza el comportamiento de la extensión propuesta de OLSR frente a ataques de diferentes características para comprobar la estabilidad del protocolo. También se comprueba la eficiencia de COD-OLSR ante diferentes porcentajes de atacantes y en situaciones diversas para ver cómo evolucionan las métricas de rendimiento más representativas como son la sobrecarga, la tolerancia a fallos del sistema, el ratio de entrega de paquetes y el *throughput*. En la simulación se han realizado las siguientes consideraciones:

- Los nodos intrusos no realizan simultáneamente ataques en los mensajes de control *Hello* y *TC*.
- El ataque en cuestión o es una suplantación de identidad o bien una suplantación de los enlaces, no considerándose ambos tipos de ataques en el mismo mensaje.

Tabla 9.3: Comparativa con otros protocolos de autoconfiguración

| **Gestión de Direcciones** | **Protocolo** | **(G)** | **(S)** | **(L)** | **(D)** | **(U)** |
|---|---|---|---|---|---|---|
| Protocolos de estado completo | ManetConf | No | Alta | Alta | No | Sí |
| | Mohsin & Prakash | Sí | Media | Media | No | No |
| | Buddy System | Sí | Media | Media | No | Sí |
| | EMAP | No | Baja | Alta | No | Sí |
| | **D2HCP** | **Sí** | **Baja** | **Baja** | **Sí** | **Sí** |
| | **E-D2HCP** | **Sí** | **Baja** | **Baja** | **Sí** | **Sí** |
| Protocolos sin estado | SDAD | No | Alta | Alta | No | Sí |
| | WDAD | No | Media | Baja | No | No |
| | PDAD | No | Media | Baja | No | Sí |
| | APAC | No | Alta | Alta | Sí | No |
| | AROD | Sí | Alta | Alta | No | No |
| | AIPAC | No | Alta | Alta | No | No |
| Protocolos híbridos | HCQA | Sí | Alta | Alta | Sí | No |
| | PACMAN | No | Alta | Alta | Sí | Sí |

### 9.8.1 Tolerancia a Fallos del Sistema

Como se observa en la Figura 9.11, la tolerancia a fallos del sistema depende más del número de intrusos que del número de mensajes atacados. Se observa que si hay un 100 % de ataques de mensajes con un 20 % de intrusos la pérdida de paquetes es algo superior al 12 %. También se observa que cuando disminuye el número de intrusos al 5 %, la pérdida de mensajes no supera el 2 % sea cual sea el porcentaje de mensajes atacados. Dado que COD-OLSR presenta una buena tolerancia a fallos, proporciona estabilidad a OLSR.

### 9.8.2 Ratio de Entrega de Paquetes de Datos

La Figura 9.12 presenta el ratio de entrega de paquetes de datos, teniendo en cuenta que los nodos maliciosos atacan los mensajes de control en un 100 % (caso extremo). Se observa que el ratio depende del aumento de la densidad de nodos: en redes poco densas (20 nodos) la línea resultante es casi uniforme manteniéndose en el 20 % de ratio, independientemente del número de atacantes; en redes muy densas (100 nodos) el ratio disminuye desde el 70 % hasta el 20 %, cuando se incrementa el número de intrusos (ya que los ataques provocan que las rutas no se hagan correctamente).

### 9.8.3 Throughput

Como se observa en la Figura 9.13, el *throughput* se comporta de forma análoga al ratio de entrega de paquetes de datos.

### 9.8.4 Sobrecarga en el Número de Paquetes

La Figura 9.14 analiza el comportamiento de la sobrecarga en el número de paquetes según la densidad de nodos de la red con el 100 % de ataques de los mensajes. En redes poco densas (20 nodos), la sobrecarga no llega a superar los 5 paquetes, presentando un valor uniforme sea cual sea el número de ataques. Cuando aumenta el número de intrusos en redes muy densas (100 nodos), la sobrecarga aumenta considerablemente. Se observa que con un 20 % de atacantes la sobrecarga alcanza un valor de más de 30 paquetes.

### 9.8.5 Sobrecarga en el Número de Bytes

Como se observa en la Figura 9.15, en redes poco densas la sobrecarga en el número de bytes es prácticamente despreciable tanto si el porcentaje de atacantes es del 0 % como si es del 20 %. En cambio, crece considerablemente en redes muy densas cuando aumenta el número de atacantes.

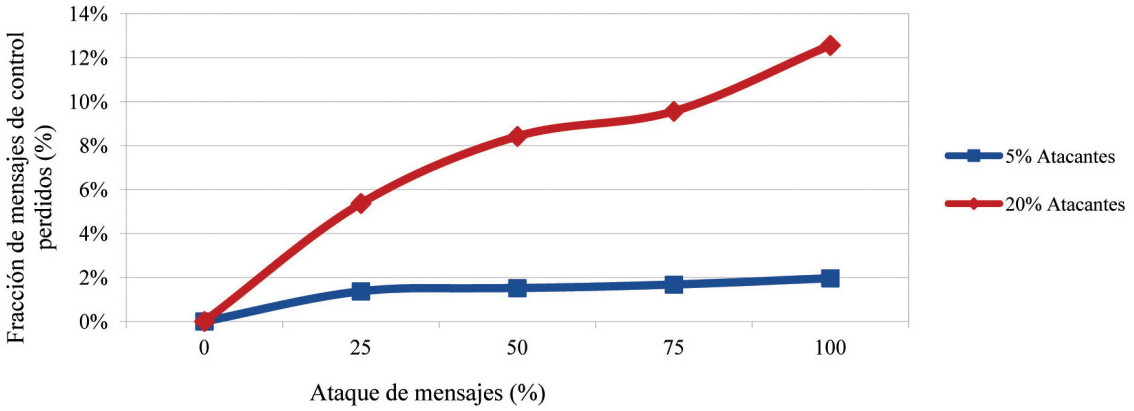



Figura 9.11: Tolerancia a fallos del sistema

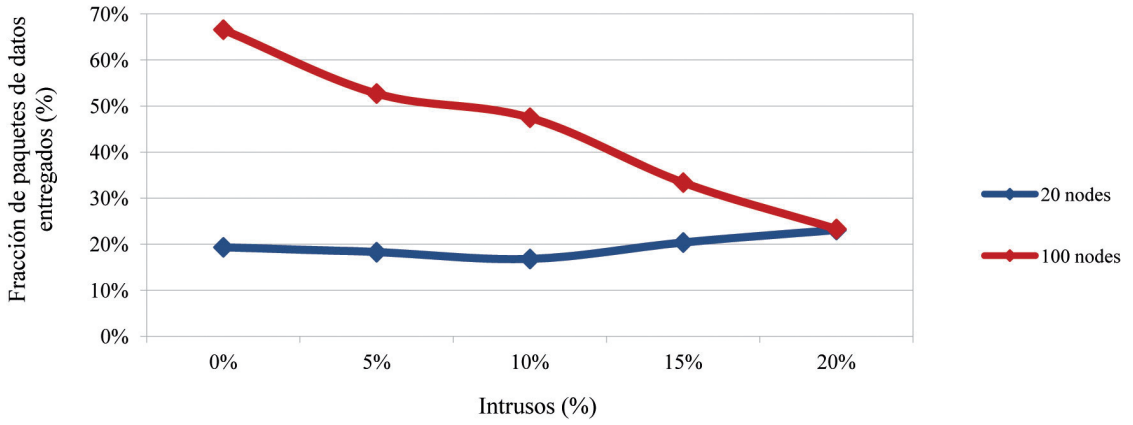



Figura 9.12: Ratio de entrega de paquetes de datos

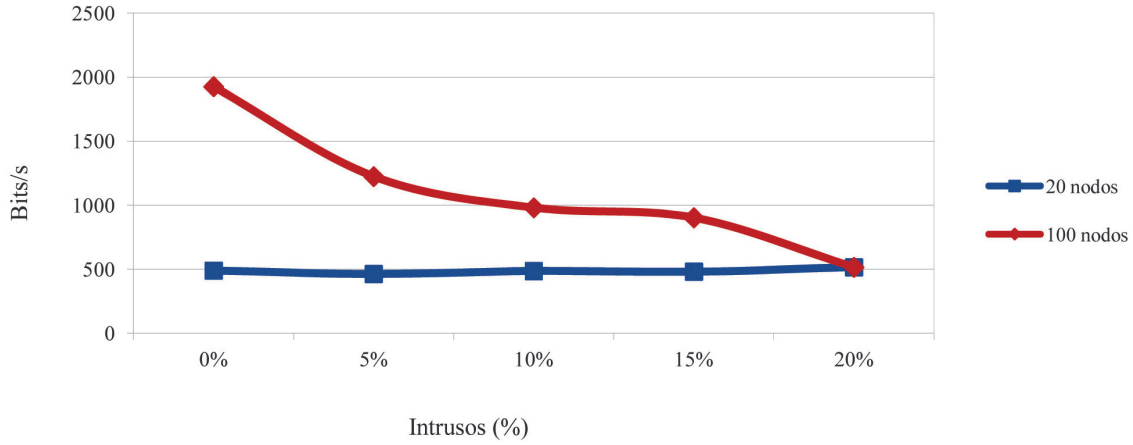



Figura 9.13: Throughput

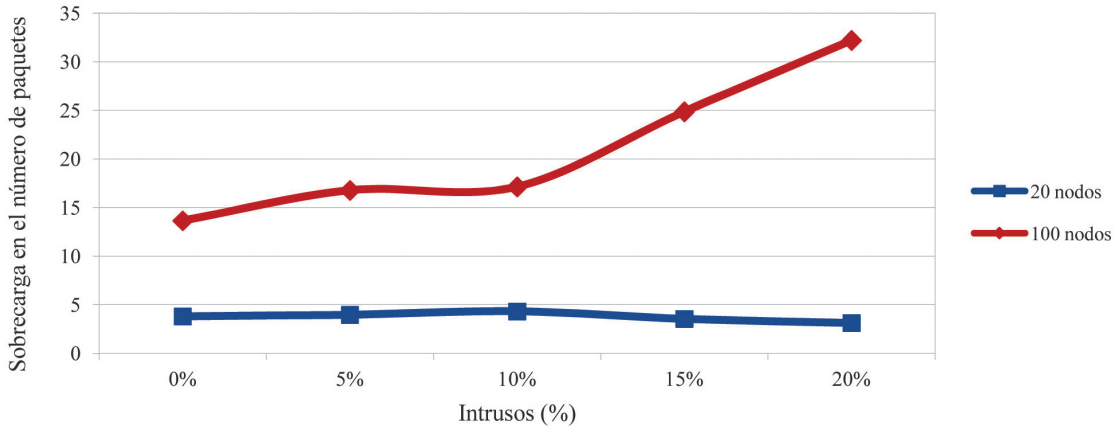



Figura 9.14: Sobrecarga en el número de paquetes

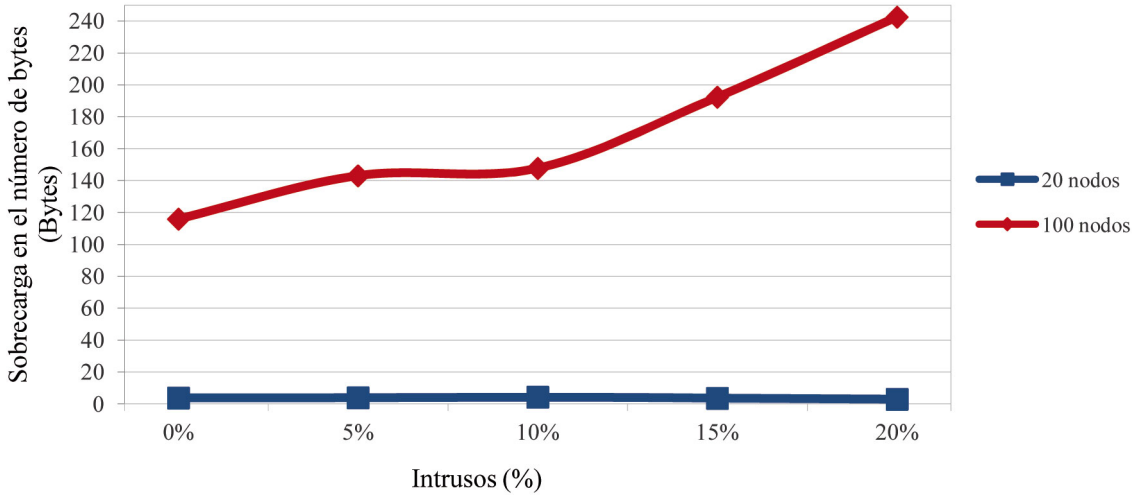



Figura 9.15: Sobrecarga en el número de bytes

## 9.9 COD-OLSR versus OLSR

En esta comparativa se ha utilizado una densidad de nodos variable entre 20 y 100, cotejándose la sobrecarga de los dos protocolos sin considerar ningún tipo de ataque para poder apreciar cómo la sobrecarga de COD-OLSR no es significativa. Se comparan tanto la sobrecarga en el número de paquetes como la sobrecarga en el número de bytes, ya que la primera tiene en cuenta el número de mensajes por paquete, mientras que la segunda hace referencia al tamaño total en bytes del paquete.

### 9.9.1 Sobrecarga en el Número de Paquetes

Como se observa en la Figura 9.16, la sobrecarga en el número de paquetes en ambos protocolos es muy similar, salvo en redes muy densas donde COD-OLSR presenta una ligera sobrecarga respecto a OLSR.

### 9.9.2 Sobrecarga en el Número de Bytes

Como se observa en la Figura 9.17, la sobrecarga en el número de bytes es ligeramnte mayor en COD-OLSR que en OLSR conforme aumenta la densidad de nodos. Esto es debido a que la cabecera de los mensajes de control de COD-OLSR tiene 3 campos más de 4 bytes cada uno, por lo que el protocolo tiene que procesar 12 bytes más.

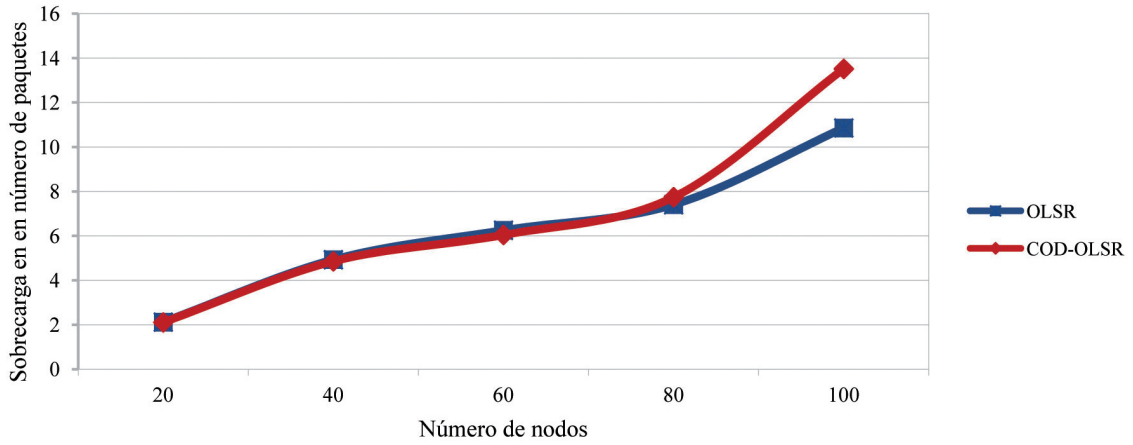



Figura 9.16: Sobrecarga en el número de paquetes

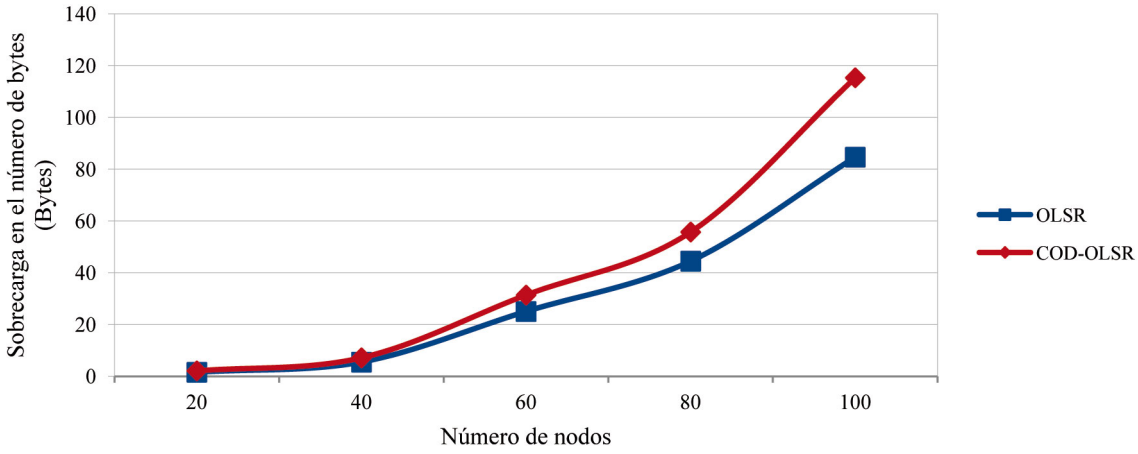



Figura 9.17: Sobrecarga en el número de bytes

## 9.10 Síntesis del Capítulo

Este capítulo ha recogido las simulaciones realizadas considerando escenarios reales con el objeto de verificar la aplicabilidad de D2HCP, E-D2HCP y COD-OLSR. Para ello se ha utilizado el simulador de redes NS-3 [NS3].

Tanto en D2HCP como en E-D2HCP se han evaluado las siguientes métricas: latencia en la asignación de direcciones, número de mensajes intercambiados, sobrecarga en el número de paquetes y sobrecarga en el número de bytes. Los resultados de las simulaciones demuestran que el protocolo de autoconfiguración D2HCP presenta baja latencia y sobrecarga, y que su extensión, E-D2HCP, presenta latencia y sobrecarga aún menores, lo que permite inferir la escalabilidad de ambos protocolos. Una comparativa de ambos protocolos ha mostrado que las diferencias entre ellos son más significativas en el caso de redes densas, esto es, en el caso de porcentajes elevados de ocupación de la red. Asimismo, ha sido realizada una comparación de los principales protocolos de autoconfiguración, observándose claramente las ventajas de D2HCP y E-D2HCP frente al resto.

En COD-OLSR se han evaluado las siguientes métricas: tolerancia a fallos del sistema, ratio de entrega de paquetes de datos, *throughput*, sobrecarga en el número de paquetes y sobrecarga en el número de bytes. Los resultados de las simulaciones demuestran que COD-OLSR añade una ligera sobrecarga a OLSR, que apenas afecta al rendimiento, validando la bondad de la citada extensión de OLSR.

# Capítulo 10

# Conclusiones y Trabajo Futuro

Este trabajo ha abordado aspectos fundamentales de las redes móviles ad hoc como son la autoconfiguración, el encaminamiento y la seguridad.

En primer lugar se ha realizado una revisión del estado del arte de los protocolos de autoconfiguración para redes móviles ad hoc observándose que no existían en la literatura protocolos representativos que consiguieran baja latencia y sobrecarga simultáneamente. A priori sólo un protocolo de autoconfiguración de estado completo puede cumplir el requisito de baja latencia, lo que focalizó la investigación al diseño de un protocolo de autoconfiguración de este tipo.

En segundo lugar se ha analizado el problema del encaminamiento en redes móviles ad hoc, pero no de una forma independiente sino intentando relacionarlo en todo momento con el diseño del protocolo de autoconfiguración. Se ha visto claramente entonces que los protocolos de encaminamiento proactivos resultan idóneos en este sentido, particularmente OLSR, al introducir mecanismos de inundación controlada que mantienen la sobrecarga en valores admisibles.

En tercer lugar se ha estudiado la problemática de la seguridad en redes móviles ad hoc, observándose las grandes deficiencias existentes en la literatura pues la inmensa mayoría de los protocolos de autoconfiguración y encaminamiento para redes móviles ad hoc están diseñados sin tener en cuenta el posible comportamiento malicioso de alguno de los nodos, lo que puede ser aprovechado para vulnerar la seguridad de la red. Además de lo anterior, y tras haber revisado los principales trabajos sobre autoconfiguración y encaminamiento seguros puede concluirse que las soluciones planteadas son muy específicas, siendo claramente una línea abierta de investigación.

Posteriormente, se ha presentado la especificación del protocolo de autoconfiguración D2HCP que gestiona la entrada y salida de nodos en redes móviles ad hoc. El protocolo hace que los nodos de una red móvil ad hoc colaboren entre sí para gestionar la asignación de direcciones IP únicas y correctas de forma distribuida. Todos los nodos de la red tienen el mismo papel, no existiendo un tipo de nodo especial que centralice la gestión de la misma. Los nodos cuentan con un sistema de sincronización que se apoya en el protocolo de encaminamiento OLSR. Gracias a este mecanismo la sincronización se lleva a cabo de forma pasiva, monitorizando el protocolo de encaminamiento mencionado, por lo que se genera una sobrecarga nula en el tráfico de la red con respecto al generado por el protocolo OLSR. Al ser todos los nodos responsables de gestionar la entrada de cualquier otro nuevo nodo a la red, esta operación puede realizarse rápidamente. Un nodo que desea entrar a una red intenta contactar con cualquier nodo ya perteneciente a ella, pudiendo recibir respuesta de varios nodos. Esto hace que las posibilidades de entrar a formar parte de la red con éxito sean altas, debido a la alta disponibilidad y a la redundancia

que supone la gestión distribuida. El protocolo desarrollado garantiza unicidad en las direcciones IP bajo una amplia variedad de condiciones de red que incluyen pérdida de mensajes, peticiones concurrentes y partición de redes. Cabe destacar asimismo la gestión óptima del direccionamiento y la escalabilidad que presenta el protocolo frente a otras propuestas existentes en la literatura y su flexibilidad que facilita la extensión del protocolo con nuevas funcionalidades.

A continuación se ha especificado E-D2HCP, una extensión del protocolo anterior más versátil: contempla la fusión de redes, resuelve la concurrencia en la inicialización de la red e incorpora nuevos mecanismos de tolerancia a fallos.

Seguidamente se ha especificado una extensión de OLSR que proporciona seguridad a OLSR ante los ataques de Generación de Mensajes Incorrectos en sus dos modalidades: Suplantación de Identidad y Suplantación de Enlaces.

Finalmente, se han realizado diversas simulaciones en NS-3 con el objetivo de validar las tres propuestas anteriores. La simulación demuestra que el protocolo de autoconfiguración D2HCP tiene baja latencia y sobrecarga, y que su extensión, el protocolo E-D2HCP, presenta latencia y sobrecarga aún menores que las de su predecesor, siendo más significativa la diferencia entre ellos en redes densas. Las diversas métricas analizadas han permitido concluir asimismo la escalabilidad de ambos protocolos. La simulación también demuestra que COD-OLSR añade una ligera sobrecarga a OLSR, que apenas afecta al rendimiento, siendo una interesante alternativa para proveer integridad en OLSR frente a los mecanismos clásicos que hacen uso de criptografía, más complejos y con una gran sobrecarga.

## 10.1 Trabajos Futuros

Aunque el protocolo de autoconfiguración y encaminamiento seguro ofrezca una solución atractiva para el diseño de una red móvil ad hoc, muchas son las cuestiones que pueden plantearse. Las principales líneas de investigación que se derivan del presente trabajo son:

- **Estudio del protocolo de autoconfiguración en cooperación con otros protocolos de encaminamiento proactivos**. Tanto D2HCP como E-D2HCP se han diseñado para trabajar de forma conjunta con OLSR. Podrían analizarse otros protocolos proactivos que pudieran desempeñar un papel similar al de OLSR.

- **Extensión del protocolo de autoconfiguración a redes subordinadas**. Tanto D2HCP como E-D2HCP son protocolos de autoconfiguración de estado completo para redes aisladas. Se podría realizar una extensión de ambos protocolos a redes subordinadas, con acceso a Internet o a otro tipo de redes, para lo que habría que tener en cuenta la topología de la red.

- **Extensión del protocolo de autoconfiguración a direcciones IPv6**. Tanto D2HCP como E-D2HCP son protocolos de autoconfiguración de direcciones IPv4 para redes móviles ad hoc. Podría hacerse una extensión de ambos protocolos para realizar el proceso de autoconfiguración de direcciones IPv6.

- **Extensión segura del protocolo de autoconfiguración**. Podría añadirse un módulo de seguridad que proteja al protocolo de autoconfiguración contra diferentes ataques para poder proporcionar una autoconfiguración segura. Ésta es, sin duda, una de las líneas más prometedoras dada la inexistencia de soluciones en la literatura especializada. Aún cobraría mayor interés si el módulo de seguridad también pudiera añadirse a otros protocolos representativos de autoconfiguración.

- **Estudio de la técnica de codificación empleada en COD-OLSR en otros protocolos de encaminamiento proactivos**. La extensión de seguridad de OLSR realizada apenas introduce sobrecarga. Podría plantearse su aplicabilidad a otros protocolos de encaminamiento proactivos.
- **Ampliación de la extensión de seguridad del protocolo de encaminamiento OLSR**. COD-OLSR proporciona integridad a OLSR ante los ataques de Generación de Mensajes Incorrectos. Podrían estudiarse ampliaciones de este esquema para prevenir otros posibles ataques a los que está expuesto OLSR.

# Bibliografía


[Abr70] N. Abramson. THE ALOHA SYSTEM - Another Alternative for Computer Communications. In *Proceedings of the Fall Joint Computer Conference*, volume 37, pages 281–286, Houston, Texas, November 1970.

[ACL+03] C. Adjih, T. Clausen, A. Laouiti, P. Mühlethaler, and D. Raffo. Securing the OLSR Protocol. In *Proceedings of the Second IFIP Annual Mediterranean Ad Hoc Networking Workshop*, pages 1–10, Mahdia, Tunisia, June 2003.

[AKG08] L. Abusalah, A. Khokhar, and M. Guizani. A Survey of Secure Mobile Ad Hoc Routing Protocols. *IEEE Communications Surveys & Tutorials*, 10(4):78–93, 2008.

[AUT] Ad Hoc Network Autoconfiguration Work Group (autoconf Work Group), http://tools.ietf.org/wg/autoconf/.

[Bac08] E. Baccelli. Address Autoconfiguration for MANET: Terminology and Problem Statement. Internet Draft draft-ietf-autoconf-statement-04, Internet Engineering Task Force, February 2008.

[BCM08a] C. Bernardos, M. Calderon, and H. Moustafa. Ad-Hoc IP Autoconfiguration Solution Space Analysis. Internet Draft draft-bernardos-autoconf-solution-space-02, Internet Engineering Task Force, November 2008.

[BCM08b] C. Bernardos, M. Calderon, and H. Moustafa. Evaluation Considerations for IP Autoconfiguration Mechanisms in MANETs. Internet Draft draft-bernardos-autoconf-evaluation-considerations-03, Internet Engineering Task Force, November 2008.

[BCM08c] C. Bernardos, M. Calderon, and H. Moustafa. Survey of IP Address Autoconfiguration Mechanisms for MANETs. Internet Draft draft-bernardos-manet-autoconf-survey-04, Internet Engineering Task Force, November 2008.

[BHvR05] R. Barr, Z. J. Haas, and R. van Renesse. *Scalable Wireless Ad Hoc Network Simulation*, chapter 19, pages 297–311. CRC Press, August 2005.

[BR05] E. M. Belding-Royer. *Mobile Ad Hoc Networking*, chapter 10, pages 275–300. Wiley Inter-Science, first edition, 2005.

[CAG05] S. Cheshire, B. Aboba, and E. Guttman. Dynamic Configuration of IPv4 Link-Local Addresses. RFC 3927, Internet Engineering Task Force, May 2005.

[CE95] M. S. Corson and A. Ephremides. A Distributed Routing Algorithm for Mobile Wireless Networks. *Wireless Networks*, 1(1):61–81, February 1995.

[CG98] T.-W. Chen and M. Gerla. Global State Routing: a New Routing Scheme for Ad-Hoc Wireless Networks. In *Proceedings of the IEEE International Conference on Communications*, volume 1, pages 171–175, San Francisco, CA, USA, June 1998.

[CO04] A. Cavalli and J.-M. Orset. Secure Hosts Auto-Configuration in Mobile Ad Hoc Networks. In *Proceedings of the Twenty-Fourth International Conference on Distributed Computing Systems Workshops*, pages 809–814, Hachioji, Tokyo, March 2004.

[CP03] T. Clausen and Jacquet P. Optimized Link State Routing Protocol (OLSR). RFC 3626, Internet Engineering Task Force, October 2003.

[CP10] I. Chakeres and C. E. Perkins. Dynamic MANET On-Demand (DYMO) Routing. Internet Draft draft-ietf-manet-dymo-21, Internet Engineering Task Force, July 2010.

[DBV+03] R. Droms, J. Bound, B. Volz, T. Lemon, C. Perkins, and M. Carney. Dynamic Host Configuration Protocol for IPv6 (DHCPv6). RFC 3315, Internet Engineering Task Force, October 2003.

[Dro97] R. Droms. Dynamic Host Configuration Protocol. RFC 2131, Internet Engineering Task Force, March 1997.

[Fee01] L. M. Feeney. An Energy Consumption Model for Performance Analysis of Routing Protocols for Mobile Ad Hoc Networks. *Mobile Networks and Applications*, 6(3):239–249, 2001.

[FL01] J. A. Freebersyser and B. Leiner. *Ad Hoc Networking*, chapter 2, pages 29–51. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, first edition, 2001.

[FVP06] M. Fazio, M. Villari, and A. Puliafito. IP Address Autoconfiguration in Ad Hoc Networks: Design, Implementation and Measurements. *Computer Networks*, 50(7):898–920, 2006.

[GHP02] M. Gerla, X. Hong, and G. Pei. Fisheye State Routing Protocol (FSR) for Ad Hoc Networks. Internet Draft draft-ietf-manet-fsr-03, Internet Engineering Task Force, July 2002.

[GVGMSO06] L. J. García-Villalba, J. García-Matesanz, and A. L. Sandoval-Orozco. Redes Ad-Hoc Seguras: Sistema de Emergencias como Caso de Uso. Informe Final Proyecto FIT-360000-2005-65, Ministerio de Industria, Turismo y Comercio (MITyC), Enero 2006.

[GVGMSO07] L. J. García-Villalba, J. García-Matesanz, and A. L. Sandoval-Orozco. Gestión de la Seguridad y Confianza en Redes Ad Hoc Híbridas. Informe Final Proyecto FIT-360000-2006-64, Ministerio de Industria, Turismo y Comercio (MITyC), Enero 2007.

[GVGMSO11] L. J. García-Villalba, J. García-Matesanz, and A. L. Sandoval-Orozco. Protocolo Seguro de Autoconfiguración de Direcciones IP para Redes Móviles Ad Hoc. Informe Final Proyecto TEC2007-67129/TCM, Ministerio de Ciencia e Innovación (MICINN), Enero 2011.

[HHW04] C. Huang, H.-P. Hu, and Z. Wang. Modeling Time-Related Trust. In *Proceedings of the Grid and Cooperative Computing Workshops*, LNCS 3252, pages 382–389, October 2004.

[HM05] S. Hu and C. Mitchell. Improving IP Address Autoconfiguration Security in MANETs Using Trust Modelling. In *Proceedings of the First International Conference on Mobile Ad-Hoc and Sensor Networks*, LNCS 3794, pages 83–92, Wuhan, China, December 2005.

[HPJ06] Y. C. Hu, A. Perrig, and D. B. Johnson. Wormhole Attacks in Wireless Networks. *IEEE Journal on Selected Areas in Communications*, 24(2):370–380, February 2006.

[HPP02] Z. J. Haas, M. R. Pearlman, and Samar P. The Zone Routing Protocol (ZRP) for Ad Hoc Networks. Internet Draft draft-ietf-manet-zone-zrp-04, Internet Engineering Task Force, July 2002.

[HRFR06] T. R. Henderson, S. Roy, S. Floyd, and G. F. Riley. NS-3 Project Goals. In *Proceedings of the Workshop on NS-2: The IP Network Simulator*, page 13, Pisa, Italy, 2006.

[HTR+04] A. Halfslund, A. Tonnesen, R. B. Rotvik, J. Andersson, and O. Kure. Secure Extension to the OLSR Protocol. In *Proceedings of the OLSR Interop and Workshop*, pages 1–4, San Diego, California, USA, October 2004.

[IET] The Internet Engineering Task Force (IETF), http://www.ietf.org/.

[Jai03] S. Jain. Energy Aware Communication in Ad - Hoc Networks. Technical Report UW-CSE 03-06-03, University of Washington, Computer Science and Engineering, Seattle, January 2003.

[JCPK03] J. Jeong, H. Cha, J. Park, and H. Kim. IP Address Autoconfiguration for Ad Hoc Networks. Internet Draft draft-jeong-adhoc-ip-addr-autoconf-00, Internet Engineering Task Force, May 2003.

[JG07] G. Jayakumar and G. Gopinath. Ad Hoc Mobile Wireless Networks Routing Protocols - A Review. *Journal of Computer Science*, 3(8):574–582, 2007.

[JHM07] D. Jonson, Y. Hu, and D. Maltz. The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4. RFC 4728, Internet Engineering Task Force, February 2007.

[KÖ8] M. M. Köksal. A Survey of Network Simulation Tools: Current Status and Future Developments, http://www.cse.wustl.edu/jain/cse567-08/ftp/simtools/index.html, November 2008.

[KAL07] N. Kim, S. Ahn, and Y. Lee. AROD: An Address Autoconfiguration with Address Reservation and Optimistic Duplicated Address Detection for Mobile Ad Hoc Networks. *Computer Communications*, 30(8):1913–1925, June 2007.

[KNK+07] B. Kannhavong, H. Nakayama, N. Kato, A. Jamalipour, and Y. Nemoto. A Study of a Routing Attack in OLSR-Based Mobile Ad Hoc Networks. *International Journal of Communication Systems*, 20(11):1245–1261, November 2007.

[KV00] Y.-B. Ko and N. H. Vaidya. Location-Aided Routing (LAR) in Mobile Ad Hoc Networks. *Wireless Networks*, 6(4):307–321, July 2000.

[LCXL07] L. Li, Y. Cai, X. Xu, and Y. Li. Agent-Based Passive Autoconfiguration for Large Scale MANETs. *Wireless Personal Communications*, 43:1741–1749, 2007.

[Liu05] J. J.-N. Liu. *Mobile Ad Hoc Networking*, chapter 1, pages 1–45. Wiley Inter-Science, first edition, 2005.

[Man] Mobile Ad-Hoc Networks Work Group (manet), http://tools.ietf.org/wg/manet/.

[MC03] L A N Man and Standards Committee. IEEE Standard for Local and Metropolitan Area Networks - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. *IEEE Std. 802.11-1999*, June 2003.

[MC05] J. P. Macker and M. S. Corson. *Mobile Ad Hoc Networking*, chapter 9, pages 255–274. Wiley Inter-Science, first edition, 2005.

[MDMD01] A. Misra, S. Das, A. McAuley, and S. K. Das. Autoconfiguration, Registration, and Mobility Management for Pervasive Computing. *IEEE Personal Communications*, 8(4):24–31, August 2001.

[MGLA95] S. Murthy and J. J. García-Luna-Aceves. A Routing Protocol for Packet Radio Networks. In *Proceedings of the First Annual International Conference on Mobile Computing and Networking*, pages 86–95, Berkeley, California, USA, 1995.

[MM00] A. J. McAuley and K. Manousakis. Self-Configuring Networks. In *Proceedings of the Twenty-First Century Military Communications Conference*, volume 1, pages 315–319, Los Angeles, California, USA, 2000.

[Moy98] J. Moy. OSPF Version 2. RFC 2328, Internet Engineering Task Force, April 1998.

[MP02] M. Mohsin and R. Prakash. IP Address Assignment in a Mobile Ad Hoc Network. In *Proceedings of the Century Military Communications Conference*, volume 2, pages 856–861, Anaheim, California, USA, October 2002.

[NA08] F. Nait-Abdesselam. Detecting and Avoiding Wormhole Attacks in Wireless Ad Hoc Networks. *IEEE Communications Magazine*, 46(4):127–133, April 2008.

[NNSS07] T. Narten, E. Nordmark, W. Simpson, and H. Soliman. Neighbor Discovery for IP version 6 (IPv6). RFC 4861, Internet Engineering Task Force, September 2007.

[NP01] S. Nesargi and R. Prakash. DADHCP: Distributed Dynamic Configuration of Hosts in a Mobile Ad Hoc Network. Technical Report UTDCS-04-01, University of Texas at Dallas, Department of Computer Science, January 2001.

[NP02] S. Nesargi and R. Prakash. MANETconf: Configuration of Hosts in a Mobile Ad Hoc Network. In *Proceedings of the Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 2, pages 1059–1068, New York, NY, USA, June 2002.

[NS2] The Network Simulator NS-2, http://www.isi.edu/nsnam/ns.

[NS3] The Network Simulator NS-3, http://www.nsnam.org/.

[OMN] Objective Modular Network Testbed in C++ - OMNET++, http://www.omnetpp.org/.

[OSP] Open Shortest Path First IGP Work Group (OSPF), http://tools.ietf.org/wg/ospf/.

[OTL04] R. Ogier, F. Templin, and M. Lewis. Topology Dissemination Based on Reverse-Path Forwarding (TBRPF). RFC 3684, Internet Engineering Task Force, February 2004.

[Pat01] P. Patchipulusu. Dynamic Address Allocation Protocols for Mobile Ad Hoc Networks. Master's thesis, Texas A. & M. University, August 2001.

[PB94] C. E. Perkins and P. Bhagwat. Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers. *SIGCOMM Computer Communications Review*, 24(4):234–244, October 1994.

[PBRD03] C. E. Perkins, E. M. Belding-Royer, and S. Das. Ad Hoc On-Demand Distance Vector (AODV) Routing. RFC 3561, Internet Engineering Task Force, July 2003.

[PC01] V. Park and S. Corson. Temporally-Ordered Routing Algorithm (TORA) Version 1 Functional Specification. Internet Draft draft-ietf-manet-tora-spec-04, Internet Engineering Task Force, July 2001.

[PH03] P. Papadimitratos and Z. J. Haas. Secure Link State Routing for Mobile Ad Hoc Networks. In *Proceedings of the Symposium on Applications and the Internet Workshops*, pages 379–383, Orlando, Florida, USA, January 2003.

[PHM+06] R. Puttini, M. Hanashiro, F. Miziara, R. de Sousa, L. J. García-Villalba, and C. J. Barenco Abbas. On the Anomaly Intrusion-Detection in Mobile Ad Hoc Network Environments. In *Proceedings of the Eleventh IFIP International Conference on Personal Wireless Communications*, LNCS 4217, pages 182–193, Albacete, Spain, 2006.

[PMW+01] C. E. Perkins, J. T. Malinen, R. Wakikawa, E. M. Belding-Royer, and Y. Sun. IP Address Autoconfiguration for Ad Hoc Networks. Internet Draft draft-perkins-manet-autoconf-01, Internet Engineering Task Force, November 2001.

[PPM+03] R. S. Puttini, J.-Marc Percher, L. Mé, O. Camp, R. De Sousa, C. J. Barenco Abbas, and L. J. García-Villalba. A Modular Architecture for Distributed IDS in MANET. In *Proceedings of the International Conference on Computational Science and its Applications: Part III*, pages 91–113, Montreal, Canada, May 2003.

[RACM04a] D. Raffo, C. Adjih, T. Clausen, and P. Mühlethaler. An Advanced Signature System for OLSR. In *Proceedings of the Second ACM Workshop on Security of Ad Hoc and Sensor Networks*, pages 10–16, Washington DC, USA, 2004.

[RACM04b] D. Raffo, C. Adjih, T. Clausen, and P. Mühlethaler. OLSR with GPS Information. In *Proceedings of the Internet Conference*, Tsukuba, Japan, 2004.

[RGLA99] J. Raju and J. J. García-Luna-Aceves. A New Approach to On-Demand Loop-Free Multipath Routing. In *Proceedings of the Eighth International Conference on Computer Communications and Networks*, pages 522–527, Boston, Massachusetts, USA, October 1999.

[RRP06] F. Ros, P. Ruiz, and C. E. Perkins. Extensible MANET Auto-Configuration Protocol (EMAP). Internet Draft draft-ros-autoconf-emap-02, Internet Engineering Task Force, September 2006.

[STC08] J. P. Sheu, S. C. Tu, and L. H. Chan. A Distributed IP Address Assignment Scheme in Ad Hoc Networks. *International Journal of Ad Hoc and Ubiquitous Computing*, 3(1):10–20, December 2008.

[TD03] O. Troan and R. Droms. IPv6 Prefix Options for DHCPv6. RFC 3633, Internet Engineering Task Force, December 2003.

[TN98] S. Thomson and T. Narten. IPv6 Stateless Address Autoconfiguration. RFC 2462, Internet Engineering Task Force, December 1998.

[TNJ07] S. Thomson, T. Narten, and T. Jinmei. IPv6 Stateless Address Autoconfiguration. RFC 4862, Internet Engineering Task Force, September 2007.

[TP06] M. R. Thoppian and R. Prakash. A Distributed Protocol for Dynamic Address Assignment in Mobile Ad Hoc Networks. *IEEE Transactions on Mobile Computing*, 5(1):4–19, January 2006.

[TT09] M. Tajamolian and M. Taghiloo. Lightweight Secure IP Address Auto-Configuration Based on VASM. In *Proceedings of the International Conference on Advanced Information Networking and Applications Workshops*, pages 176–180, University of Bradford, Bradford, UK, May 2009.

[Vai02] N. H. Vaidya. Weak Duplicate Address Detection in Mobile Ad Hoc Networks. In *Proceedings of the Third International Symposium on Mobile Ad Hoc Networking & Computing*, pages 206–216, Lausanne, Switzerland, June 2002.

[Wen03] K. Weniger. Passive Duplicate Address Detection in Mobile Ad Hoc Networks. In *Proceedings of the IEEE Wireless Communications and Networking Conference*, volume 3, pages 1504–1509, Paris, France, March 2003.

[Wen05] K. Weniger. PACMAN: Passive Autoconfiguration for Mobile Ad Hoc Networks. *IEEE Journal on Selected Areas in Communications*, 23(3):507–519, March 2005.

[WRN05] P. Wang, D. S. Reeves, and P. Ning. Secure Address Auto-Configuration for Mobile Ad Hoc Networks. In *Proceedings of the Second Annual International Conference on Mobile and Ubiquitous Systems*, pages 519–522, Los Alamitos, CA, USA, 2005.

[WvLW09] E. Weingärtner, H. vom Lehn, and K. Wehrle. A Performance Comparison of Recent Network Simulators. In *Proceedings of the IEEE International Conference on Communications*, Dresden, Germany, June 2009.

[WZ02] K. Weniger and M. Zitterbart. IPv6 Autoconfiguration in Large Scale Mobile Ad-Hoc Networks. In *Proceedings of the European Wireless Conference*, pages 142–148, Florence, Italy, February 2002.

[WZ04] K. Weniger and M. Zitterbart. Mobile Ad Hoc Networks - Current Approaches and Future Directions. *IEEE Network*, 18(4):6–11, July 2004.

[YSBR03] E. Yuan Sun and M. Belding-Royer. Dynamic Address Configuration in Mobile Ad Hoc Networks. Technical Report UCSB 2003-11, University at Santa Barbara, Department of Computer Science, June 2003.

[ZMN09] H. Zhou, M. W. Mutak, and L. M. Ni. Secure Autoconfiguration and Public-Key Distribution for Mobile Ad-Hoc Networks. In *Proceedings of the IEEE Sixth International Conference on Mobile Ad Hoc and Sensor Systems*, pages 256–263, Macau (S.A.R.), China, October 2009.

[ZNM03] H. Zhou, L. M. Ni, and M. W. Mutka. Prophet Address Allocation for Large Scale MANETs. In *Proceedings of the Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 2, pages 1304–1311, San Francisco, CA, USA, March 2003.

# Parte II

# Publicaciones

# Apéndice A

# Lista de Publicaciones

A continuación se muestran las publicaciones que han resultado de la realización de esta Tesis Doctoral

- Luis Javier García Villalba, Julián García Matesanz, Ana Lucila Sandoval Orozco, José Duván Márquez Díaz. Auto-Configuration Protocols in Mobile Ad Hoc Networks. *Sensors*, 11:3652–3666, March 2011.
- Luis Javier García Villalba, Julián García Matesanz, Ana Lucila Sandoval Orozco, José Duván Márquez Díaz. Distributed Dynamic Host Configuration Protocol (D2HCP). *Sensors*, 11:4438-4461, April 2011.
- Julián García Matesanz, Luis Javier García Villalba, Ana Lucila Sandoval Orozco, José René Fuentes Cortez. An Improved Buddy System Auto-Configuration Protocol for Mobile Ad Hoc Networks. *Proceedings of the 5th International Conference on Information Technology (ICIT 2011)*, Amman, Jordan, May 11 - 13, 2011.
- Luis Javier García Villalba, Julián García Matesanz, Delfín Rupérez Cañas, Ana Lucila Sandoval Orozco. Secure Extension to the Optimised Link State Routing Protocol. *IET Information Security*, 5(3):163-169, September 2011.
- Luis Javier García Villalba, Julián García Matesanz, Ana Lucila Sandoval Orozco, José René Fuentes Cortez. An Extension Proposal of D2HCP for Network Merging. *Proceedings of the Third International Conference on Emerging Ubiquitous Systems and Pervasive Networks (EUSPN 2011)*, Amman, Jordan, October 10-13, 2011. *Journal of Ubiquitous Systems & Pervasive Networks*, 3(1):35-40, October 2011.
- Luis Javier García Villalba, Julián García Matesanz, Ana Lucila Sandoval Orozco, Delfín Rupérez Cañas, Tai-hoon Kim. E-D2HCP: Enhanced Distributed Dynamic Host Configuration Protocol. Aceptado en *Special Issue of the International Conference on Computer and Applications*, 2012.
- Ana Lucila Sandoval Orozco, Julián García Matesanz, Luis Javier García Villalba, José Duván Márquez Díaz, Tai-hoon Kim. Secure Auto-Configuration Protocols in Mobile Ad Hoc Networks. Aceptado en *Special Issue of the International Conference on Computer and Applications*, 2012.

*Sensors* **2011**, *11*, 3652-3666; doi:10.3390/s110403652

OPEN ACCESS

*sensors*

**ISSN 1424-8220**

www.mdpi.com/journal/sensors

*Review*

# Auto-Configuration Protocols in Mobile *Ad Hoc* Networks

**Luis Javier García Villalba [1,*], Julián García Matesanz [2], Ana Lucila Sandoval Orozco [1,3] and José Duván Márquez Díaz [3]**

[1] Grupo de Análisis, Seguridad y Sistemas (GASS), Departamento de Ingeniería del Software e Inteligencia Artificial (DISIA), Facultad de Informática, Despacho 431, Universidad Complutense de Madrid (UCM), Calle Profesor José García Santesmases s/n, Ciudad Universitaria, 28040 Madrid, Spain; E-Mail: asandoval@fdi.ucm.es (A.L.S.O.)

[2] Grupo de Análisis, Seguridad y Sistemas (GASS), Sección Departamental de Sistemas Informáticos y Computación—Lenguajes y Sistemas Informáticos y Ciencias de la Computación e Inteligencia Artificial, Facultad de Ciencias Matemáticas, Despacho 310-F, Universidad Complutense de Madrid (UCM), Plaza de Ciencias, 3, Ciudad Universitaria, 28040 Madrid, Spain; E-Mail: julian@sip.ucm.es (J.G.M.)

[3] Departamento de Ingeniería de Sistemas, Universidad del Norte, Km. 5 Autopista a Puerto Colombia, Barranquilla, Colombia; E-Mails: alsandoval@uninorte.edu.co (A.L.S.O.); jmarquez@uninorte.edu.co (J.D.M.D.)

[*] Author to whom correspondence should be addressed; E-Mail: javiergv@fdi.ucm.es; Tel.: +34-913-947-638; Fax: +34-913-947-547.

*Received: 5 February 2011; in revised form: 12 March 2011 / Accepted: 22 March 2011 / Published: 25 March 2011*

---

**Abstract:** The TCP/IP protocol allows the different nodes in a network to communicate by associating a different IP address to each node. In wired or wireless networks with infrastructure, we have a server or node acting as such which correctly assigns IP addresses, but in mobile *ad hoc* networks there is no such centralized entity capable of carrying out this function. Therefore, a protocol is needed to perform the network configuration automatically and in a dynamic way, which will use all nodes in the network (or part thereof) as if they were servers that manage IP addresses. This article reviews the major proposed auto-configuration protocols for mobile *ad hoc* networks, with particular emphasis on one of the most recent: D2HCP. This work also includes a comparison of auto-configuration protocols for mobile *ad hoc* networks by specifying the most relevant metrics, such as a guarantee of uniqueness, overhead, latency, dependency on the routing protocol and uniformity.

**Keywords:** auto-configuration protocol; Mobile *Ad Hoc* Network (MANET); IP address assignment; distributed dynamic host configuration protocol; D2HCP, routing protocol

## 1. Introduction

A *Mobile Ad hoc NETwork* (MANET) is a set of mobile nodes which communicate between themselves through wireless links. In contrast with conventional networks, a MANET does not need any previous infrastructure, since nodes rely on each other to operate themselves, forming what is called multi-hop communication.

Such networks have more problems and disadvantages than a conventional network. The topology of mobile networks may change quickly and in an unpredictable way. Moreover, variations in the capacity of nodes and links, frequent transmission errors and a lack of security may occur. Finally, the limited resources of the nodes must be taken into account given that an *ad hoc* network will normally be formed by battery operated devices.

To communicate with each other [1] the *ad hoc* nodes need to configure their interfaces with local addresses which are valid within the *ad hoc* network. The *ad hoc* nodes may also need to set global routing addresses to communicate with other devices on the Internet. From the perspective of the IP layer, an *ad hoc* network presents itself as a multi-hop level 3 network constituted by a collection of links.

This paper is organized into six sections, including this introduction. Section 2 tackles the problematic implications of the design of auto configuration protocols for mobile *ad hoc* networks. Section 3 discusses the inapplicability of the standard solutions. In Section 4, a classification of auto-configuration protocols for mobile *ad hoc* networks is carried out, with the most representative being mentioned, and special emphasis being made on a protocol proposed by the author: D2HCP. Then, in Section 5, a comparative study of the above-mentioned protocols is performed. Finally, Section 6 of the paper draws together the conclusions of this piece of work.

## 2. Auto-Configuration in Mobile *Ad Hoc* Network

The nodes of a network need some mechanism to interchange messages with each other. The TCP/IP protocol allows the different nodes from the network to communicate by associating a distinct IP address to each node of the same network. In wired or wireless networks with an infrastructure, there is a server or node which correctly assigns these IP addresses.

Mobile *ad hoc* networks, on the other hand, do not have such a centralized entity able to carry out this function. Therefore, some protocol that performs the network configuration in a dynamic and automatic way is necessary, which will utilize all the nodes of the network (or only part of them) as if they were servers which manage IP addresses.

Due to the dynamic topology of mobile *ad hoc* networks (constant movement of the nodes that can join and leave the network frequently and even simultaneously), auto-configuration protocols are faced with various problems in guaranteeing the uniqueness of IP addresses and in allowing network partitioning and merging.

To guarantee the correct functioning of the network, the protocols strive to achieve the following objectives [2-4]:

- *Assign unique IP addresses*: Ensure that two or more nodes do not obtain the same IP address.
- *Function correctly*: An IP address is only associated with a node for the time that it is kept in the network. When a node leaves the network, its IP address should then became available for association to another node.
- *Fix the problems derived from the loss of messages*: In case of any node failure or if message loss occurs, the protocol should operate quick enough to prevent two or more nodes from having the same IP address.
- *Allow multi-hop routing*: A node will not be configured with an IP address if there aren't any available in the whole network. Thus, if any node of the network has a free IP address, it has to associate itself with the node which is requesting an IP address, even though it is at two-hops of distance or more.
- *Minimize the additional packet traffic in the network*: The protocol must minimize the number of packets exchanged among the nodes in the auto-configuration process. In other words, control packets traffic must cause as little harm as possible to the data packet traffic, given that in the extreme case, the network performance would decrease.
- *Verify the existence of competing petitions for an IP address*: When two nodes request an IP address at the same time, the protocol must carry out the pertinent treatment so that the same IP address is not given to two nodes.
- *Be flexible to partitioning and merging of the mobile ad hoc network*: The protocol must be able to achieve the union of two different mobile *ad hoc* networks as well as the possible partitioning into two networks.
- *Conduct synchronization*: The protocol must adapt itself to the rapid changes of the wireless network topology due to the frequent mobility of the nodes. The synchronization is carried out periodically to ensure the configuration of the network is as up to date as possible.

## 3. Applicability of Standard Solutions

The applicability of the standard protocols is insufficient for MANET [5]. Two of these protocols are presented next.

### *3.1. SLAAC/NDP*

*StateLess Address AutoConfiguration* (SLAAC) [6] is a standard that allows the automatic auto-configuration of an IPv6 address without the need of a router node. For this, it needs help from the protocol *Neighbour Discovery Protocol* (NDP) [7], a standard to transmit the messages and to discover its neighbours. A node automatically creates an IPv6 address joining its *host* identifier (it is usually the MAC address) with a local well-known prefix and undergoes a DAD process by *broadcasting* NDP messages to the neighbours.

If the IPv6 address is not unique, the auto-configuration process will stop and it will be necessary to do it manually. If on the contrary the address is unique, it will have to ask for the network prefix

through NDP messages and, then, if its IPv6 address is effectively unique, it will verify with DAD again.

The applicability of this protocol in mobile *ad hoc* networks is limited because it uses the NDP protocol to send the messages and NDP assumes that all the nodes are connected to each other in the network. Consequently, it only supports one-hop transmission, whereas mobile *ad hoc* networks are most frequently multi-hop, thus not reaching the majority of the nodes to carry out the DAD processes and not being able, therefore, to ensure that the obtained IPv6 address is unique.

### *3.2. DHCP-PD*

*Dynamic Host Configuration Protocol – Prefix Delegation* (DHCP-PD) [8] is an option derived from DHCPv6 [9] which provides a mechanism for the delegation of the IPv6 address prefixes and lets one of these be assigned automatically. For this, if a node wishes to get an IPv6 address, it sends a DHCP message with the activated *prefix delegation* option to obtain a prefix from a DHCP server in the network.

Its applicability is limited in mobile *ad hoc* networks because it is based on DHCP, so it assumes that all the nodes can connect to a DHCP server, whether directly or through several hops, and due to the MANET topology, the direct connection to DHCP server is not usually frequent with the consequence that the connection through several hops can make the server unreachable.

## 4. Classification of Auto-Configuration Protocols

The auto-configuration protocols may be classified according to address management:

- *Stateful*: The nodes know the network state, *i.e.*, they keep tables with the IP addresses of the nodes.
- *Stateless*: The IP address of a node is managed by itself. Generally they create a random address and perform a process of duplicated address detection steps to verify their uniqueness.
- *Hybrid Protocols*: They mix mechanisms from the previous ones to improve the scalability and reliability of the auto-configuration. Their algorithms have a high level of complexity.

### *4.1. Stateful Protocols*

#### 4.1.1. MANETConf

MANETConf [10], which is an improvement of [11], is based on existence of a common distributed table so that all the nodes are able to assign IP addresses. When a node wants to join the network, it sends broadcast messages to other nodes and the first one which replies to the message, chooses it as an initiator node and it can supply an IP address. The initiator node chooses one of the free IP addresses located in the network and before assigning it, asks for permission from the rest of the nodes, because it is possible that another node may have also chosen this address or the tables might not be totally synchronized due to message delays. If the answer from the nodes is positive, it assigns the IP address to the joining node and it communicates this action by *broadcasting* so that the rest of the nodes keep their tables updated. If there is a node which does not reply, it is put in direct contact with

it (*unicast*) to get an answer. If it still cannot get it, it will assume that the node has left the network, and will communicate this to the rest of the nodes in the network in order to keep the tables updated.

#### 4.1.2. DAAP

Dynamic Address Allocation Protocol (DAAP) [12] is based on the concept of address assignment by a leader. The leader functionality is shared among all network nodes. When a new node joins the network, it becomes the leader until the next node joins. The leader maintains the highest IP address within the *ad hoc* network and a unique identifier is associated with the network.

Each node stores the highest IP address, which is that of the leader, and periodically sends HELLO messages to its neighbours. These HELLO messages include the network identifier so that any merging and partitioning can be detected. When a node receives a HELLO message with a different network ID, merging is detected, if a node does not receive the message that contains the current network ID, then after a timeout, a partition is detected.

#### 4.1.3. Moshin and Prakash's Protocol

The Moshin and Prakash proactive scheme [13] tries to fix the problem of IP address assignment by binary division of free address blocks. These divisions are done to the power of 2. Thus, nodes can exist with any block. Moreover, each node of the network has a table with the state of all the others in the network; that is to say, it knows the free address blocks of other nodes, as well as the IP address of the network interface from each node.

The address assignment process can come from any node. Following this idea, a non-configured node (client node) sends a *broadcast* message of the REQUEST type so that a node in the network configures this client node. As it is a diffusion message, it is possible to have several replies, but it will choose the node that replies first (REPLAY), and this will act as the server node. If this node has several blocks, it will give one to the client node, and will choose the first IP address from the block as its own. Otherwise, if the node has a block, then it will divide into two similar parts and will give one half to the client node, and keep the other one for itself.

If we consider a case in which there aren't any blocks available, some solutions are proposed based on the idea that the server searches its surrounding neighbours to see whether any block is available. In the case where no free addresses are found, it will try to get a block by using the neighbors of a higher level hop, and so on. Another solution consists of looking for the node that has the greater free range address, to deliver half of this block to the node which is joining the network.

The departure of the nodes can take place in an abrupt or smooth way, and for each case there is a scheme to follow. If the departure is abrupt, the node that acted as the server in its configuration will see in its routing table that the block of node that left the network is not there and it will add the block to its free address block. In the case of easy departure, the node, which leaves the network, warns it's neighbours of its intention to leave and the neighbour node then seeks the node which configured the client so that it gets its block.

The greatest advantages of this protocol is that it works well for merging and partitioning of networks, since it fixes the duplicated addresses problem which occurs in these cases. Each node which initiates the network generates a random number called *PartitionID* which will be a network

identification number. In this manner, when a partitioning or merging of the network occurs, the first node which leaves from original network will create another *PartitionID*. At the time the two networks with different PartitionID are joined; firstly the consistency of its IP address will be checked. This process consists of verifying the existence of two nodes with the same address. In case affirmative, a change of the node belonging to the network with less free IP address range is produced. The new IP address will belong to the highest network range with the biggest free address number. The major drawback of this protocol is that the synchronization depends on the existence of a reliable *broadcast* and such a thing does not exist in a distributed mobile environment, thus one can question the robustness of this protocol.

#### 4.1.4. Thoppian and Prakash's Protocol

An improvement of the previous scheme (particularly in terms of synchronization) can be found in [14], where Thoppian and Prakash propose a dynamic address assignment based on a so-called *buddy system* that manages mobility of nodes during address assignment, message loss, network partitioning and merging. However, the IP address allocation can generate a high overhead of control messages while it does a global search and the address recovery (to avoid missing addresses) requires diffusion messages by a *flooding* process. In addition, union and partition may incur in high overhead because of the global nature of this protocol.

#### 4.1.5. EMAP

*Extensible Manet Auto-configuration Protocol* (EMAP) [15] is an auto-configuration protocol based on the idea of a protocol of REQUEST/REPLAY messages. The main advantage of this protocol is the possibility of doing it extensibly, *i.e.*, it can include new functionalities in the future that are analyzed in a theoretical way, such as *Domain Name Server* (DNS).

This protocol also considers the possibility of exterior communications to the mobile *ad hoc* network via Internet. The route discovery mechanism among nodes is similar to the *Ad Hoc On-Demand Distance Vector* (AODV) [16] protocol.

The main idea of this auto-configuration protocol consists in having three different addresses for a non-configured node that wants to join the created network. These are three addresses for the interior communications: *temporary address*, *tentative address* and *mobile* ad hoc *network local address*.

When a node wishes join the network, it randomly generates two valid IP addresses (with network known addresses), considering them as temporary and tentative addresses. These IP addresses are encapsulated into a *Detection Adress Detection REsPonse* (DAD_REP) message to know whether it is a valid address. The node keeps waiting for a *Detection Address Detection REQuest* (DAD_REQ) message. If time runs out for this message, the node assumes that it can use its tentative address as unique, and assigns it to its network interface. If this node receives a DAD_REP message to its temporary address and this message contains the source with the tentative address that had been proposed, the node knows that this tentative address is being used and the previous process begins creating another pair of addresses again.

4.1.6. Sheu *et al.*' s Protocol

The auto-configuration protocol introduced in [17] proposes a scheme where the nodes are classified into coordinator and common nodes. The first coordinator assigned to initiate the IP address assignment is the so-called *C-root*. The coordinators manage the IP address pool and they are responsible for assigning an IP address to a node which has just joined the network. The nodes that wish to join the network will interchange *HELLO* messages to find the coordinator node nearest and to obtain a new IP address from that coordinator node. To maintain the IP address pool efficiently, the coordinator nodes are distributed in a tree topology called *C-tree* by exchanging *HELLOs*. This protocol does not consider network partition and merging.

4.1.7. D2HCP

The Distributed Dynamic Host Configuration Protocol (D2HCP) [18] is an auto-configuration protocol for mobile *ad hoc* networks which derives from improvements made to the Mohsin and Prakash's [13] protocol which guarantee the uniqueness of IP addresses used in the network. The protocol makes the nodes of a MANET work together to manage the unique and correct IP address assignment in a distributed manner. All network nodes have the same role; there is no special node type that centralizes the management.

The protocol is based on the OLSR routing protocol [19] to perform synchronization, and thus detect changes in the network. This synchronization procedure produces a null overhead in network traffic compared to that generated by the OLSR protocol. In this scheme, when a node enters the network, as well as assigning a valid IP address, a range of consecutive addresses are given. From that point on, the new node is responsible for managing the range of addresses, and can allocate part of its addresses to other nodes requesting entry into the network.

To request a new IP address, the requesting node, referred to as client, sends a *Server_Discovery* message to its neighbours. The neighbour nodes that are part of the network will answer this request by sending a *Server_Offer* message. The client node will choose one of the received responses and will send a *Server_Poll* message to the chosen neighbour node requesting an address. The node which will assign the new address will receive the server name. When the client node receives the *Server_Poll* message, it will divide its address range into two halves, and give the second half of the client. If the addresses provided by the server node were not their own, but a third node of the network, the server sends an *IP_Range_Request* message formally requesting these addresses to the node that owns the address range. The third network node sends an *IP_Range_Return* message authorizing the node that sent the *IP_Range_Request* message to assign the address block to the client node. After receiving the *Server_Poll* message, if the provided addresses were from the server node, or if *IP_Range_Return* message was necessary in the case of having had to ask for address to a third node, the node server sends an *IP_assigned* message to the client with assigned range.

Design Considerations:

- Each node has as associated information, a single block of contiguous free IP addresses for auto-configuration, which include the own IP address of the node itself.

- The responsibility of recover the IP addresses that a node makes available when it leaves the network is one that can join this free block to the right of its own. This is not possible when the block to be collected contains the lowest address of the network. In that case, the node that collects the block is one that can add to it on the left.
- By dividing the free addresses in two blocks to deliver one of them to a new node entering the network, the node that makes the client server delivers sub-block does not contain its own IP address.

### *4.2. Stateless Protocols*

#### 4.2.1. Process of Duplicated Address Detections

*Duplicate Address Detection* (DAD) is a process which uses the protocols to check the uniqueness of IP addresses. This process takes a relatively long time to complete, so several solutions have been implemented to reduce it. There are three kinds of DAD processes:

- *Strong Duplicate Address Detection (SDAD)* [20]: Is the base of the Stateless protocols. It consists of a simple mechanism whereby the node chooses two IP addresses: temporary and tentative. It will only use the temporary address for the initialization while it detects if the tentative one is unique or not. The detection method consists of sending a message ICMP destined directly to this address. If it receives a response, this IP address is being used so the process will be resumed. If it does not receive a response, the message will be sent a certain number of times to make sure that the address is unique. By being a very simple mechanism, it does not ensure the uniqueness of the IP address since the process limits itself to only the phase of initialization, and it would not work for temporary disconnections or losing of the network. Moreover, when the network is long and only a few free IP addresses remain, it increases the overhead until it finds a unique IP address.
- *Weak Duplicate Address Detection (WDAD)* [21]: It establishes the idea of tolerating the duplicated address in the network for a period of time. For that, every node when it is being initiated itself will create a key that it will always send along with its IP address. When a node receives a message, it will check whether this IP address is already assigned in its table and will look whether the keys coincide, if they do not coincide, it will mark that address as invalid and actions will be taken so that they are unique (these actions are not defined in WDAD). This process has to support the identification of a node by means of a key-IP pair and depends completely on the routing protocol. It will only work with the proactive one that updates the routes constantly, but with a reactive one there will be nodes that could never detect the duplicity of IP addresses. It does not add additional overhead to the routing protocol, but on the other hand, it really adds to the overhead by sending always the key along with the IP address.
- *Passive Duplicate Address Detection (PDAD)* [22]: The idea is based on sending control information instead of detecting or solving duplicated IP addresses, every node investigates and deduces whether a duplicated address exits by events that would never happen if all the IP addresses were unique.

Three passive detections are proposed, which are necessary for correct functioning of the detection:

- *Sequence Numbers (PDAD-SN)*: This system is based on the idea that the routing protocols use sequence numbers in its messages to update the routes. Using these sequence numbers, and the idea that two nodes with a distance between them of two-hops do not have the same neighborhood, some conflicts are solved. Also, it has taken into consideration the possibility that these sequence numbers reach the maximum and numbers start from zero again.
- *Locality Principle (PDAD-LP)*: Lower power than the previous one, is based on the frequency of updating the route tables. On the basis of this frequency it can detect duplicate addresses by taking a time threshold to display the status of the route tables. It's necessary to take into account the protocol routing used. We must consider different thresholds, since it can be the case that two messages with the same source are confused with a duplicate address, if the time is too short and, therefore, the protocol modifies the routes too fast.
- *Neighborhood (PDAD-NH)*: Taking into account that a node knows its neighbours, and they have sent a package of the state of its link, it differs if there is conflict or not depending on whether it is in a package, the source of this message is a neighbour, and contains the address. The advantage is that it does not add overhead to network, but it can only be used with proactive routing protocols.

#### 4.2.2. APAC

*Agent based Passive Auto-configuration* (APAC) [23] is an auto-configuration protocol based on PDAD. Its main feature is the use of certain nodes which centralize the distribution of addresses. The mechanism by which a node configures its IP address upon entering the network consists of asking if it has some node type *Address Agent* (AA) within a one hop distance. In that case, the AA node will give it an IP address.

If it does not receive a response from any AA node, the incoming node is configured to operate in AA mode, and it will be a server of addresses for the next incoming nodes. When it is configured as AA, the node randomly generates an identifier number *agentID*, in order to form a table with the addresses that it will assign to the nodes that arrive. These addresses have the form *agented* + *hostID*.

When a node is moved in the network and leaves the radio coverage of AA proportionately to its IP address, this node must ask for another address from another node AA within its new radio coverage. This is complemented by a mechanism preventing communication interruptions in the process. In this situation, the previous AA will mark its IP address as free in order to be allocated to some other node later.

The detection of duplicated addresses is undertaken in the PDAD process. Once any conflict has been detected, the AA which assigned the conflictive address is informed. This AA node will then generate a new *agentID* and it will warn all its dependant nodes to change their address from *agented* + *hostID* type to the new agentID.

With regards to the partitioning and merging of networks, the same above mentioned mechanism is utilized. In case of partitioning, the nodes AA will mark addresses which have left the network as free. In the case of the merging of two networks, the mechanism for the detection of duplicate addresses continues working properly.

#### 4.2.3. AROD

In *Address auto-configuration with address Reservation and Optimistic duplicated address Detection* (AROD) [24], the address reservation is based on the existence of nodes that have an IP address reserved to deliver it to the new nodes that enter. Two types of nodes will exist:

- Agents type 1 with a reserved IP address, apart from the IP address that has its network interfaces. When a node joins the network, this reserved IP will be assigned to it immediately.
- Agents type 2, which do not have reserved IP addresses. If a node that joins newly asks one of these for an IP address, this node borrows the reserved address of one of its neighbours who is of type 1, and it is assigned to the new one immediately.

It establishes a mechanism so that the network does not remain without nodes of type 1, whenever an IP address is assigned to a new node, the node that has delivered the address generates two random IP addresses (one for itself and another one for the new node that has joined) and does a DAD process to detect if these addresses are unique. Once the process has been carried out the following possibilities can arise:

- The IP addresses are unique; therefore, the two nodes turn into node type 1.
- If only one is unique, the one that gave the IP address will be turned into node type 1.
- If none is unique, the two nodes will remain as type 2.

This protocol considers its process of duplicated address detection as an optimistic DAD process, because it is only carried out once when a node joins and a reserved IP address is assigned to it. It contemplates the possibility of changing the reserved IP address number to the node type 1. If this number increases, the latency of IP address assignment is minor, but on the contrary the overhead increases due to having had to undergo more DAD processes, and *vice versa*. This possibility allows swapping latency *versus* overhead.

#### 4.2.4. AIPAC

*Automatic IP Address Configuration in Mobile* Ad Hoc *Networks* (AIPAC) [25] is a protocol for IP address auto-configuration using a reactive approach in the IP address assignment, so it should manage duplicate addresses; it is also focused on maintaining the address uniqueness after network merging due to the node mobility. The protocol has as its priority the supporting of the following features of *ad hoc* networks: limited resources of the devices and the unreliability in the wireless channels.

Each network is identified with its *NetID*. When two networks merge, and the merger is persistent, the *NetID* should be unified. To accomplish that, it uses a gradual fusion mechanism. This allows a node to pass from a *NetID* to another one, according to network changes observed by the node. This procedure allows a homogeneous system to be made in the case of multiple overlapping networks, according to the evolution of their topologies.

This protocol does not guarantee the uniqueness of the assigned IP addresses, but it ensures that messages are routed correctly. Each node in AIPAC is aware of its neighbouring radio, so the amount of information stored by the node is limited to the nodes within the radios distance.

### *4.3. Hybrid Protocols*

#### 4.3.1. HCQA

*Hybrid Centralized Query-based Autoconfiguration* (HCQA) [26] was the first hybrid auto-configuration protocol. A node that wants to join the network undergoes a SDAD process. If the process is successful, the node will have to register its tentative IP address with an *Address Authority*. To do this, it will expect a message from the Address Authority and when that message has been received, it will send a registry request and the Address Authority will confirm it. The node starts a counter as this process begins, if the timer expires, it will start the process again until it can register the IP address. When the network is created, the first node becomes an Address Authority, it chooses a unique identifier for the network (e.g., MAC address) and advertises it periodically by *broadcast* messages to identify the network. If a node does not receive it, it is assumed that the network has been divided and it will create its own network becoming Address Authority. This protocol adds robustness to the SDAD process, it ensures no duplicity of IP addresses and it also provides a good mechanism for network *partitioning*. However, it has two main problems, firstly the overhead produced by the SDAD process and periodic messages of Address Authority, and secondly, the network depends on a central entity with which all nodes must communicate directly in order to register its IP address, so that much latency is added at the joining of nodes to the network.

#### 4.3.2. PACMAN

*Passive Autoconfiguration for Mobile* ad hoc *Networks* (PACMAN) [27] is a passive auto-configuration protocol for MANET. It uses elements from stateless and stateful protocols, so it could be considered somewhat hybrid. Its operation is based on each node assigning itself an address when joining the network, and passive monitoring of communications for the duplicate address detection.

To achieve the minimum overhead in the communications, the information is shared among different network layers. Specifically, the information handled by the routing protocol is monitored.The method used to choose the own IP address consists of a probabilistic algorithm. The probability of attempting to choose an IP address currently in use by another node is close to zero, this algorithm takes into account, among other factors, an assignment table. This table is created with information from the routing protocol on IP addresses already in use.

PACMAN uses the PDAD process to monitor the communications in search of duplicated addresses. This is necessary because the mechanism used for address assignment does not guarantee uniqueness (even if it attempts to reduce the probability of collision), and it may cause merging of networks containing nodes with the same IP address. Broadly speaking there are two types of events that indicate duplicity of IP addresses: firstly, we have the events that never occur if the address is unique, and always occur if the address is duplicated. These events confirm that there is a problem detected. On the other hand, we have the events that occur rarely if the address is unique, and often if

the address is duplicated. In this way the possibility of problems is detected, so it is probabilistic algorithms. When it detects that two nodes are using the same IP address, it reports the problem to one of them using a *unicast* message to change its address. Moreover, it takes into account the problem of changing an address that has some communication going on. To fix this, when changing an address, a node notifies the nodes with which it has ongoing communications of its new IP address, so that they can make an encapsulation of the messages properly.

## 5. Performance Evaluation of Auto-Configuration Protocols

### *5.1. Performance Metrics for the Evaluation of Auto-Configuration Protocols*

Zhou *et al.* [28] define some parameters that can be used to analyze the performance of an auto-configuration protocol for *ad hoc* networks (see Table 1).

**Table 1.** Evaluation metrics.

| Metrics | Description |
|---|---|
| Uniqueness | Each MANET node must have a unique IP address for each network interface because duplicate addresses can cause serious routing problems. |
| Overhead | Exchanged packet number to obtain an IP address. |
| Latency | Node timeout to obtain the IP address. |
| Routing Protocol Independence | Auto-configuration protocols can work in two ways: leaning on a routing protocol to allow the routing of the new nodes joining the network, or regardless of routing algorithm. |
| Uniformity | All nodes perform the same function in the auto-configuration process. |

### *5.2. Performance Evaluation of Studied Protocols*

The submitted protocols share some common characteristics. However, they also differ in a wide range of issues. Table 2 presents a comparison of the characteristics of IPv4 addressing protocols.

**Table 2.** Comparison of the characteristics of IPv4 addressing protocols.

| | Protocol | Guarantee of Uniqueness | Overhead | Latency | Dependent Routing | Uniform |
|---|---|---|---|---|---|---|
| **Stateful** | ManetConf | No | High | High | No | Yes |
| | DAAP | Yes | Medium | Medium | No | No |
| | Buddy System | Yes | Medium | Medium | No | Yes |
| | EMAP | No | Low | High | No | Yes |
| | D2HCP | Yes | Low | Low | Yes | Yes |
| **Stateless** | SDAD | No | High | High | No | Yes |
| | WDAD | No | Medium | Low | No | No |
| | PDAD | No | Medium | Low | No | Yes |
| | APAC | No | High | High | Yes | No |
| | AROD | Yes | High | High | No | No |
| | AIPAC | No | High | High | No | No |
| **Hybrid** | HCQA | Yes | High | High | Yes | No |
| | PACMAN | No | High | High | Yes | Yes |

## 6. Conclusions

The nodes of a network need a mechanism to exchange messages. The TCP/IP protocols can allow the different nodes of the same network to be associated with a different IP address. Due to the dynamic topology of mobile *ad hoc* networks (constant movement of the nodes that can enter and leave the network frequently or even simultaneously), auto-configuration protocols face many problems with guaranteeing the uniqueness of IP addresses. This work presents a classification of auto-configuration protocols for the most representative mobile *ad hoc* networks in literature with special emphasis on D2HCP protocol, and in particular in its design considerations and the advantages over their predecessors, especially when efficiently managing the IP address space of the *ad hoc* wireless network.

## Acknowledgements

This work was supported by the Ministerio de Industria, Turismo y Comercio (MITyC, Spain) through the Project AVANZA I+D+I COMPETITIVIDAD TSI-020100-2010-482 and by the Ministerio de Ciencia e Innovación (MICINN, Spain) through the Projects TEC2007-67129/TCM and TEC2010-18894/TCM. This work was also supported by the Departamento Administrativo de Ciencia, Tecnología e Innovación (COLCIENCIAS, Colombia) through Programa de Recuperación Contingente which funds the Project 121545221101. Ana Lucila Sandoval Orozco is also supported by the Programme Alβan, the European Union Programme of High Level Scholarships for Latin America, scholarship No. E07M403236CO.

## References

1. *Ad-Hoc* Network Autoconfiguration Work Group (autoconf). Available online: http://tools.ietf.org/wg/autoconf/ (accessed on 25 November 2010).
2. Bernardos, C.; Calderon, M.; Moustafa, H. Ad-Hoc *IP Autoconfiguration Solution Space Analysis*; Internet Draft; November 2008. Available online: http://tools.ietf.org/pdf/draft-bernardos-autoconf-solution-space-02.pdf (accessed on 25 November 2010).
3. Bernardos, C.; Calderon, M.; Moustafa, H. *Survey of IP Address Autoconfiguration Mechanisms for MANETs*; Internet Draft; November 2008. Available online: http://tools.ietf.org/html/draft-bernardos-manet-autoconf-survey-04 (accessed on 25 November 2010).
4. Bernardos, C.; Calderon, M.; Moustafa, H. *Evaluation Considerations for IP Autoconfiguration Mechanisms in MANETs*; Internet Draft; November 2008. Available online: http://www.it.uc3m.es/cjbc/papers/draft-bernardos-autoconf-evaluation-considerations-03.txt (accessed on 25 November 2010).
5. Baccelli, E. *Address Autoconfiguration for MANET: Terminology and Problem Statement*; Internet Draft; February 2008. Available online: http://tools.ietf.org/html/draft-ietf-autoconf-statement-04 (accessed on 25 November 2010).
6. Thomson, S.; Narten, T.; Jinmei, T. *IPv6 Stateless Address Autoconfiguration*; RFC 4862; September 2007. Available online: http://www.ietf.org/rfc/rfc4862.txt (accessed on 25 November 2010).

7. Narten, T.; Nordmark, E.; Simpson, W.; Soliman, H. *Neighbor Discovery for IP version 6 (IPv6)*; RFC 4861; September 2007. Available online: http://www.ietf.org/rfc/rfc4861.txt (accessed on 25 November 2010).
8. Troan, O.; Droms, R. *IPv6 Prefix Options for DHCPv6*; RFC 3633; December 2003. Available online: http://www.ietf.org/rfc/rfc3633.txt (accessed on 25 November 2010).
9. Droms, R.; Bound, J.; Volz, B.; Lemon, T.; Perkins, C.; Carney, M. *Dynamic Host Configuration Protocol for IPv6 (DHCPv6)*; RFC 3315; July 2003. Available online: http://www.ietf.org/rfc/rfc3315.txt (accessed on 25 November 2010).
10. Nesargi, S.; Prakash, R. MANETconf: Configuration of Hosts in a Mobile *Ad Hoc* Network. In *Proceedings of IEEE INFOCOM 2002*, New York, NY, USA, June 2002; pp. 1059-1068. Available online: http://www.utdallas.edu/~ravip/papers/infocom2002.pdf (accessed on 25 November 2010).
11. Nesargi, S.; Prakash, R. *DADHCP: Distributed Dynamic Configuration of Hosts in a Mobile* Ad Hoc *Network*; Technical Report UTDCS-04-01; University of Texas at Dallas, Department of Computer Science: Dallas, TX, USA, January 2001.
12. Patchipulusu, P. *Dynamic Address Allocation Protocols for Mobile* Ad Hoc *Networks*. Master's Thesis, Texas A&M University, Dallas, TX, USA, August 2001.
13. Mohsin, M.; Prakash, R. IP Address Assignment in a Mobile *Ad Hoc* Network. In *Proceedings of Military Communications Conference (MILCOM)*, Anaheim, CA, USA, September 2002; Volume 2, pp. 856-861. Available online: http://www.utdallas.edu/~ravip/papers/milcom02.pdf (accessed on 25 November 2010).
14. Thoppian, M.R.; Prakash, R. A Distributed Protocol for Dynamic Address Assignment in Mobile *Ad Hoc* Networks. *IEEE Trans. Mob. Comput.* **2006**, *5*, 4-19.
15. Ros, F.; Ruiz, P.; Perkins, C.E. *Extensible MANET Auto-Configuration Protocol (EMAP)*; Internet Draft; March 2006. Available online: http://tools.ietf.org/html/draft-ros-autoconf-emap-02 (accessed on 25 November 2010).
16. Perkins, C.E.; Belding-Royer, E.M.; Das, S. Ad Hoc *On-Demand Distance Vector (AODV) Routing*; RFC 3561; July 2003. Available online: http://tools.ietf.org/html/rfc3561 (accessed on 25 November 2010).
17. Sheu, J.P.; Tu, S.C.; Chan, L.H. A Distributed IP Address Assignment Scheme in *Ad Hoc* Networks. *Int. J. Ad Hoc Ubiquitous Comput*. **2008**, *3*, 10-20.
18. García Villalba, L.J.; García Matesanz, J.; Sandoval Orozco, A.L.; Márquez Díaz, J.D. Distributed Dynamic Host Configuration Protocol (D2HCP). *Sensors* **2011**, submitted.
19. Clausen, T.; Jacquet, P. *Optimized Link State Routing Protocol (OLSR)*; RFC 3626; October 2003. Available online: http://www.ietf.org/rfc/rfc3626 (accessed on 25 November 2010).
20. Perkins, C.E.; Malinen, J.T.; Wakikawa, R.; Belding-Royer, E.M.; Sun, Y. *IP Address Autoconfiguration for* Ad Hoc *Networks*; Internet Draft; November 2001. Available online: http://tools.ietf.org/html/draft-perkins-manet-autoconf-01 (accessed on 25 November 2010).
21. Vaidya, N.H. Weak Duplicate Address Detection in Mobile *Ad Hoc* Networks. In *Proceedings of ACM MobiHoc 2002*, Lausanne, Switzerland, June 2002; pp. 206-216.

22. Weniger, K. Passive Duplicate Address Detection in Mobile *Ad Hoc* Networks. In *Proceedings of IEEE WCNC 2003*, New Orleans, LA, USA, March 2003. Available online: http://www.tm.uka.de/doc/2003/passive_dad_lsr_wcnc03.pdf (accessed on 25 November 2010).
23. Li, L.; Cai, Y.; Xu, X.; Li, Y. Agent-Based Passive Autoconfiguration for Large Scale MANETs. *Wirel. Personal Commun.* **2007**, *43*, 1741-1749.
24. Kim, N.; Ahn, S.; Lee, Y. AROD: An Address Autoconfiguration with Address Reservation and Optimistic Duplicated Address Detection for Mobile *Ad Hoc* Networks. *Comput. Commun.* **2007**, *30*, 1913-1925.
25. Fazio, M.; Villari, M.; Puliafito, A. IP Address Autoconfiguration in *Ad Hoc* Networks: Design, Implementation and Measurements. *Comput. Netw.* **2005**, *50*, 898-920.
26. Sun, Y.; Belding-Royer, E.M. *Dynamic Address Configuration in Mobile* Ad Hoc *Networks*; Technical Report UCSB 2003-11; Department of Computer Science, University at Santa Barbara: Santa Barbara, CA, USA, June 2003.
27. Weniger, K. PACMAN: Passive Autoconfiguration for Mobile *Ad hoc* Networks. *IEEE J. Sel. Areas Commun.* **2005**, *23*, 507-519. Available online: http://www.tm.uka.de/doc/2004/autoconf_jsac_epub.pdf (accessed on 25 November 2010).
28. Zhou, H.; Lionel, M.N.; Mutka, M.W. Prophet Address Allocation for Large Scale MANETs. In *Proceedings of IEEE INFOCOM 2003*, San Francisco, CA, USA, March 2003.

*Sensors* **2011**, *11*, 4438-4461; doi:10.3390/s110404438

OPEN ACCESS

*sensors*

**ISSN 1424-8220**

www.mdpi.com/journal/sensors

*Article*

# Distributed Dynamic Host Configuration Protocol (D2HCP)

**Luis Javier García Villalba [1,*], Julián García Matesanz [2], Ana Lucila Sandoval Orozco [1,3] and José Duván Márquez Díaz [3]**

[1] Grupo de Análisis, Seguridad y Sistemas (GASS), Departamento de Ingeniería del Software e Inteligencia Artificial (DISIA), Facultad de Informática, Despacho 431, Universidad Complutense de Madrid (UCM), Calle Profesor José García Santesmases s/n, Ciudad Universitaria, 28040 Madrid, Spain; E-Mail: asandoval@fdi.ucm.es

[2] Grupo de Análisis, Seguridad y Sistemas (GASS), Sección Departamental de Sistemas Informáticos y Computación—Lenguajes y Sistemas Informáticos y Ciencias de la Computación e Inteligencia Artificial, Facultad de Ciencias Matemáticas, Despacho 310-F, Universidad Complutense de Madrid (UCM), Plaza de Ciencias, 3, Ciudad Universitaria, 28040 Madrid, Spain; E-Mail: julian@sip.ucm.es

[3] Departamento de Ingeniería de Sistemas, Universidad del Norte, Km. 5 Autopista a Puerto Colombia, Barranquilla, Colombia; E-Mail: jmarquez@uninorte.edu.co

* Author to whom correspondence should be addressed; E-Mail: javiergv@fdi.ucm.es; Tel.: +34-913-947-638; Fax: +34-913-947-547.

*Received: 28 February 2011; in revised form: 10 April 2011 / Accepted: 12 April 2011 / Published: 18 April 2011*

---

**Abstract:** Mobile *Ad Hoc* Networks (MANETs) are multihop wireless networks of mobile nodes without any fixed or preexisting infrastructure. The topology of these networks can change randomly due to the unpredictable mobility of nodes and their propagation characteristics. In most networks, including MANETs, each node needs a unique identifier to communicate. This work presents a distributed protocol for dynamic node IP address assignment in MANETs. Nodes of a MANET synchronize from time to time to maintain a record of IP address assignments in the entire network and detect any IP address leaks. The proposed stateful autoconfiguration scheme uses the OLSR proactive routing protocol for synchronization and guarantees unique IP addresses under a variety of network conditions, including message losses and network partitioning. Simulation results show that the protocol incurs low latency and communication overhead for IP address assignment.

**Keywords:** Mobile *Ad Hoc* Network (MANETs); IP address assignment; autoconfiguration; dynamic host configuration; stateful protocol; synchronization; OLSR proactive routing protocol

## 1. Introduction

A Mobile *Ad hoc* NETwork (MANET) is a set of mobile nodes which communicate through wireless links. In contrast with conventional networks, a MANET does not need a previous infrastructure, since nodes rely on each other to operate themselves, forming what is called multi-hop communication. Such networks have several disadvantages that a conventional network does not present: the topology of this kind of network may change quickly and in an unpredictable way. Moreover, variations in the capacity of nodes and links, and frequent transmission errors and lack of security could occur. Finally, the limited energy resources of the nodes must be taken into account, since normally an *ad hoc* network will be formed by devices powered by batteries.

To communicate with each other [1], the *ad hoc* nodes need to configure their interfaces with local addresses which are valid inside an *ad hoc* network. The *ad hoc* nodes may also need to set routing addresses globally to communicate with other devices on the Internet. From the perspective of the IP layer, an *ad hoc* network is presented as a multi-hop network of level 3 constituted by a collection of links.

In an autonomous *ad hoc* mobile network the nodes can be uniquely identified by an IP address with the only premise that this address must be different from that any other node in the network. The configuration process is the set of steps through which a node obtains its IP address within the network. There are two mechanisms to set addresses: *Stateless* and *Stateful*.

The *Stateless* address configuration proposes its own node to be the one in charge of generating its IP address. The address is obtained from the concatenation of a well-known network prefix and the theoretically unique number inside the network generated by the node. This mechanism may require the inclusion of a module responsible for verifying the uniqueness of the generated address called *Duplicate Address Detection* (DAD) [2-4].

On the other hand, *Stateful* address configuration is based on using servers which control and assign addresses to all the nodes of the network. The well known *Dynamic Host Configuration Protocol* (DHCP) [5] is an example of *Stateful* configuration. However, because of the multi-hop nature of mobile *ad hoc* networks, this protocol cannot be applied directly.

This work proposes a *Stateful*-based auto-configuration protocol that guarantees the uniqueness of IP addresses under a wide variety of network conditions such as missing messages and network partitioning. This work is structured in five sections; the first one is the present Introduction. Section 2 shows the obligated references in the auto-configuration protocol scope of Mobile *Ad Hoc* Networks. Section 3 contains an itemized specification of the so-called *Distributed Dynamic Host Configuration Protocol* (D2HCP), a proposal concerning IP addresses auto-configuration for MANETS. Section 4 presents the D2HCP protocol simulations carried out in NS-3 [6]. Finally, Section 5 discusses the main advantages of the newly developed protocol as well as potential future extensions to the study.

## 2. Related Works

Mobile *Ad Hoc* Networks (MANETS) present special features which must be born in mind when an address configuration protocol is implemented. There are many solutions for conventional networks (e.g., RFCs 3315 [5], 4861 [7], 4862 [8] and so on) but Mobile *Ad Hoc* Networks were not taken into account in their design. It is necessary, therefore, to provide support for multi-hop communication, dynamic topologies and the merging and partitioning of networks, events that are typical in Mobile *Ad Hoc* Networks.

There are numerous works that present proposals for address configuration in a Mobile *Ad Hoc* Network using the *Stateless* and *Stateful* mechanism. Without doubt, the most representative are those described in [2,9-21]. Bernardos *et al.* [22-24] carried out a rigorous study of the problems of the auto-configuration in Mobile *Ad Hoc* Networks, presenting an itemized review of the more representative auto-configuration protocols. A comprehensive review of the main auto-configuration protocols can be found in [25].

The *Internet Engineering Task Force* (IETF) [26] includes what is perhaps the best known work group in this field, the so-called *Ad Hoc Network Autoconfiguration Work Group* (Autoconf WG) [1] whose principal purpose is to describe the addressing model for *ad hoc* networks and how the nodes can set their addresses in these networks. It is essential that such models do not cause problems to other components of an *ad hoc* system such as standard applications which are executed in an *ad hoc* node or Internet nodes connected to the *ad hoc* nodes. The work of this group can include the development of new protocols if the existing IP auto-configuration mechanisms turn out to be inadequate. Nevertheless, the first task of this work group is to describe a practical addressing model for *ad hoc* networks.

The solutions described previously represent significant contributions to aid our comprehension of the problem, but we consider that all these approaches only handle a subset of the network conditions enumerated as follows:

(1) Dynamic Topology: the nodes in the network can move arbitrarily and may join and leave the network dynamically.
(2) Message loss and failure in the nodes: message loss can be quite frequent and can result in duplicate IP address allocation if it is not managed correctly. The nodes can abruptly depart from the network due to a link failure or an accident.
(3) *Partitioning* and *merging*: the network can split into multiple networks and, later, join with others. During network merging it is possible to have duplicated IP addresses in the fused network.
(4) Address concurrent requests: multiple nodes may want to join the network simultaneously.
(5) Limited Energy and Bandwidth: the nodes in a Mobile *Ad Hoc* Network have limited energy and the links have a limited bandwidth, therefore, the communication overhead which is incurred should be low.

In this work a solution similar to DAAP [27,28] and to the one given in [29] that guarantees uniqueness in the IP address allocation under a wide set of network conditions is proposed. In our approach, the majority of address allocations imply local communication, thus causing low communication overhead and low latency.

## 3. D2HCP (Distributed Dynamic Host Configuration Protocol)

The Distributed Dynamic Host Configuration Protocol (D2HCP) is an auto-configuration protocol that manages the addition and departure of nodes in a MANET. The protocol makes the MANET nodes collaborate with each other to manage the assignment of unique and correct IP addresses in a distributed manner. All the network nodes have the same role; there is no special type of node that centralizes the management of the same.

Nodes have a synchronization system based on the OLSR [30] routing protocol. Thanks to this mechanism, the synchronization is done passively, by monitoring the mentioned routing protocol, thus no network traffic overhead is generated compared to that generated by the OLSR protocol.

Due to the fact that all the nodes are responsible for managing the addition of any new node to the network, this process can be done quickly. A node that wishes to join a network tries to contact any node still belonging to it, and may receive several responses from multiple nodes. This makes the chances of successfully joining the network high, because of the high availability and redundancy that distributed management provides.

Here we introduce the D2HCP specification: it begins with the data structures used, continues with an explanation of the messages exchanged between nodes for joining and departing the network, and then details how synchronization takes place in the protocol. Finally, we explain the format of the messages exchanged during the auto-configuration process, detailing how to solve the problem of possible message loss in the network using appropriate timers and performing certain actions when they expire to restore the auto-configuration process, as well as state diagrams for each operation mode that a node can adopt.

### *3.1. Data Structures*

The data structures of this protocol can be classified into those handling the auto-configuration mechanism and those belonging to the OLSR routing protocol. OLSR stores internally a routing table which is updated periodically. This table contains information about the route to each node, stored in the following fields:

- *R_dest_addr*: IP address of the destination node.
- *R_next_addr*: IP address of next hop in the route.
- *R_dist*: Distance to the destination node.
- *R_iface_addr*: IP address of the outgoing interface to the destination node.

The structures necessary for auto-configuration mechanism are:

- IP addresses of the node interfaces.
- Netmask.
- Free_IP_Blocks: A table of free block from each node in the network. It will have the following form:

| IP | Free_IP_blocks |
|---|---|
| .1 | .1–.64 |
| .128 | .128–.254 |
| .65 | .65–.127 |

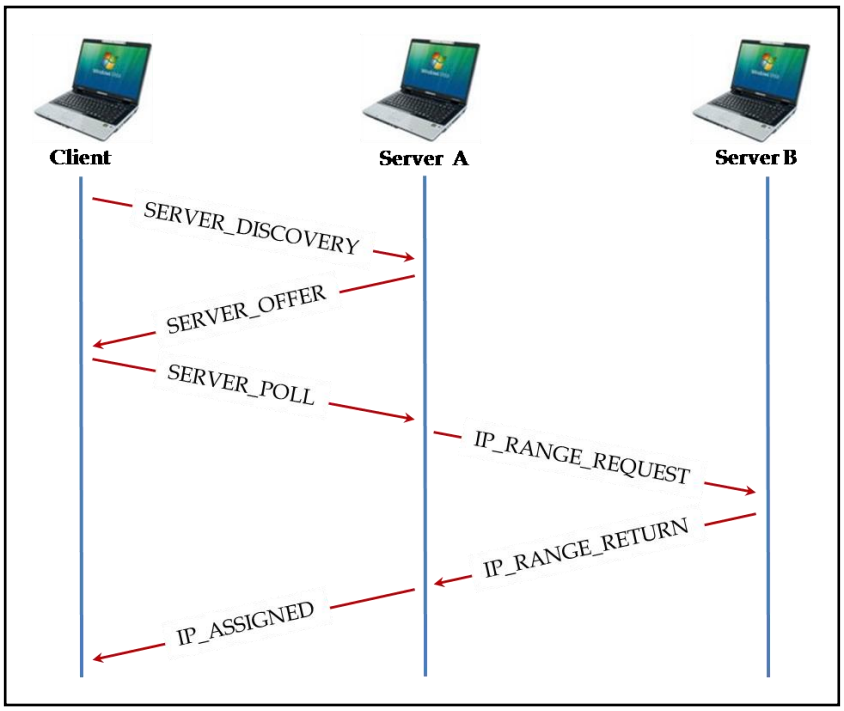
### *3.2. Joining and Departure of Nodes*

The protocol uses a specific message number for each operation. All the operations are defined seeking optimum operation and low latency. This section discusses how communication is established between the nodes and the messages transmitted during the joining and departure of nodes in the network.

#### 3.2.1. Node Joining

The entry of a node to the network implies the need to locate a node acting as a server. Once found, it will facilitate the joining by providing an IP address block and a *Free_IP_Blocks* table representing the state of all the nodes in the network. Until the node has an assigned IP address, its communication with nodes which might act as servers will be through the MAC layer. The configuration mechanism uses four types of messages in most cases. If no nodes in range with free IP addresses are found six types of messages in total will be used. Figure 1 shows the exchanged message scheme as is explained below:

**Figure 1.** The messages interchanged in the process of a node joining the network.




1. *SERVER_DISCOVERY:* The client node wishing to join a network starts the process with a message of this type. It is transmitted by the MAC layer, with the *broadcast* address as its destination. The message indicates the IP address number which is required (equal to the

interface number). If the node has more than one network interface, the message is transmitted through all of them, using the ID field, thus the different interfaces are not *confused* with several nodes.

2. *SERVER_OFFER:* The network nodes receiving the SERVER_DISCOVERY message reply to this message, also using the MAC layer, in which an IP address number is offered. The number of addresses offered is half of the available range. The SERVER_DISCOVERY message includes a *Count* field indicating how many attempts have been made by the client. Depending on its value, the server nodes will behave as follows:
   - *Count* = 1: The server node will respond with a SERVER_OFFER if enough addresses are available and the fields R (*Ready*) and L (*Local*) have the value 1 (it can assign the addresses provided in the moment, and they are addresses from the node's own block).
   - *Count* = 2: The node server will respond with a SERVER_OFFER if the fields can take the value R = 1 and L = 1. If not possible, it will still also respond if it is the case that there are enough addresses and R = 0, L = 1 (the server cannot assign addresses at the moment, but it has them).
   - *Count* > 2: If the node has addresses available and is in a state to do so, it will send a SERVER_OFFER with R = 1, L = 1. If it can, will send it with R = 0, L = 1. Finally, if it does not have enough free addresses, it will send the message with the fields R = 1, L = 0 (immediate availability of addresses, but the offered addresses are from another node in the network).
3. *SERVER_POLL:* After a certain listening time, the client node will have received several SERVER_OFFER messages. If not, it will try again. It will sort received messages using the following criteria:
   - The servers which are not available are discarded, *i.e.*, with R = 0. The SERVER_OFFER with R = 0 is not used to reply with a SERVER_POLL, but they have the function of informing the client that there is a server node in the network, although it cannot provide access to it at this moment.
   - Priority is given to local addresses: it will prefer messages with the field L = 1.
   - Finally, it is organized so that the offered addresses are ranked, from highest to lowest.

   According to this criteria for order preference, it will send a SERVER_POLL message to the first server (via the MAC layer, again) to let it know that the node has chosen this one to assign a free IP address block to it.
4. *IP_RANGE_REQUEST:* If the addresses provided by the server node were not their own, but they were from a third node in the network, with this message there will be a formal request made to that node. Since there is communication between two nodes already configured correctly, it is performed at the IP layer.
5. *IP_RANGE_RETURN:* The third network node authorizes the node that sends the message IP_RANGE_REQUEST to assign the address block indicated in this message to client nodes. It is also a message sent by IP.
6. *IP_ASSIGNED:* After receiving the SERVER_POLL, if the provided addresses were from the server node's own ones, or after an IP_RANGE_RETURN message if it was necessary to

request the address from a third node, the node server sends this message to the client. This message is transmitted by the MAC layer. In this message the free address block which is assigned to the client and *Free_IP_Blocks* table representing the network state are indicated. The table which is transmitted in this message does not reflect the joining of the client node.

After this message exchange, the client node chooses the first one of the block which has been assigned as its IP address. In the case of having more than one network interface, it will use the first ones of block in order, and will be the first of all which use as the primary address that identify the node.

#### 3.2.2. Node Departure

The node departure mechanism does not require the exchange of any messages. The node that wants to leave the network does not have to notify any other node of its departure, avoiding the overhead that these messages cause. The other nodes in the network will become aware of the departure node through the periodic route updates that the OLSR protocol performs every so often. They will note that they have lost the path to that node, and therefore they remove it from their *Free_IP_Blocks* table, adding its free address block to the corresponding node as explained in the previous section.

### *3.3. Synchronization*

The synchronization is done by monitoring the routing table of the OLSR routing protocol [30]. The addition or departure of a node in the network is detected when OLSR adds a new route to its routing table, or deletes an existing one. By detecting the addition or departure of a node in the network, the *Free_IP_Blocks* table is updated locally, and without exchanging any messages.

For this reason, the following rules are obeyed:

- The responsibility of recovering the IP addresses that a node leaving the network makes available is one that can be attached to the right of the free block. This will not be possible when the block to be collected contains the lowest address of the network. In that case, the node that picks the block up is one that can add to it to the left.
- By dividing the free addresses in two blocks to deliver one of them to a new node that joins the network, the node that acts as server delivers the sub-block, which does not contain its own IP address, to the client.

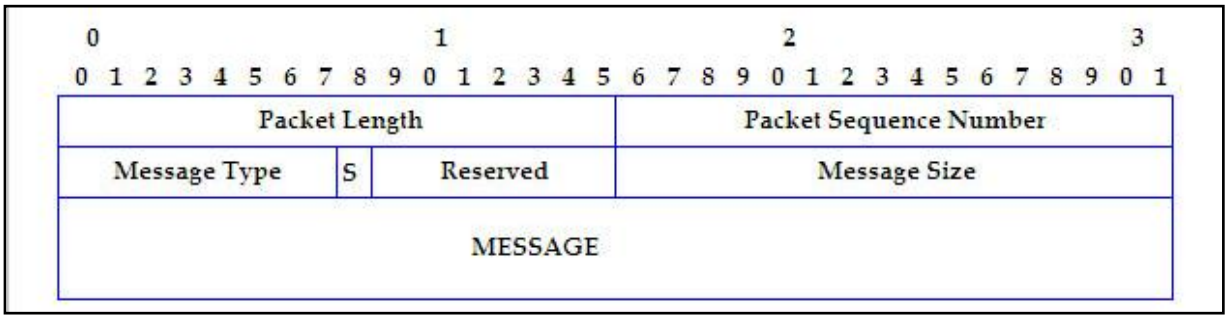
When the node departure is detected, its entry must be removed, and an update of the corresponding node's now available IP address must be recorded.

By detecting the joining of a new node, a new entry is created for it in the table, and the free address block of the node which supplied its IP address will be updated. In order to identify the node who acted as a server, it is simply necessary to find out which node has the IP address of the new node in its free address block.

### *3.4. Message Format*

All the sent messages are packed in the protocol with the format shown in Figure 2.

**Figure 2.** Packet from the D2HCP protocol.




These messages will be encapsulated in turn with the headers corresponding to the MAC or TCP/IP, depending on the type of message to be included in the MESSAGE field.

#### 3.4.1. Packet Header

The first row of the Figure 2 contains the fields of the packet header.

1. *Packet Length*: Packet length, including the header (2 bytes).
2. *Packet Sequence Number*: Sequence Number (2 bytes). In each different message which is sent by node, this field is increased by one. It helps in being able to detect duplicated packets.

#### 3.4.2. Message Header

The second row of the Figure 2 is the header of each one of the protocol message:

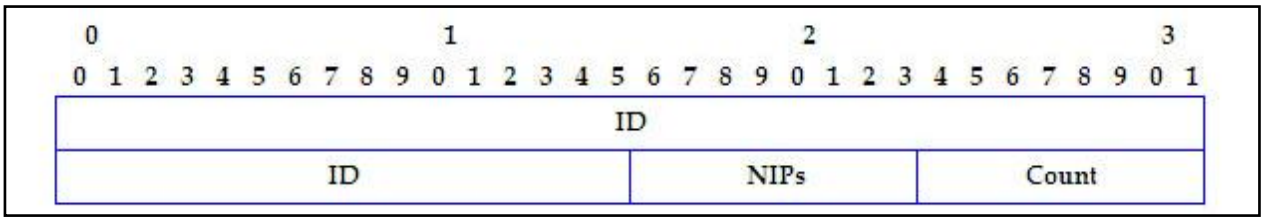
1. *Message Type*: It has 1 byte of size.
2. *S (Security)*: Reserved for security implementation (1 bit).
3. *Reserved*: Reserved for functionality future extensions (7 bits).
4. *Message Size*: It consists of 2 bytes.

Next the format of each kind of message include in the MESSAGE field is shown.

- *SERVER_DISCOVERY*

Figure 3 shows the SERVER_DISCOVERY message format.

**Figure 3.** SERVER_DISCOVERY message format.



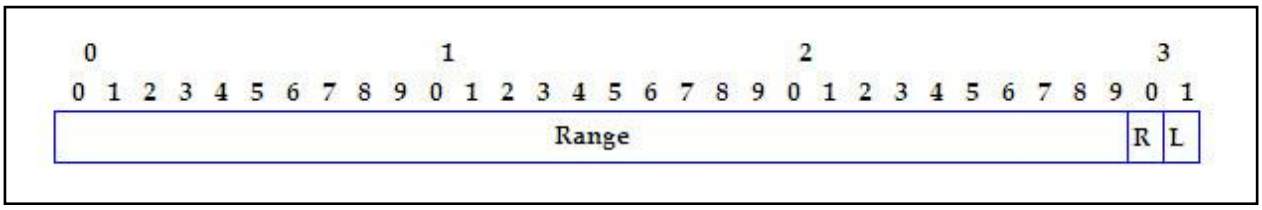
*ID*: Node Identification (6 bytes).The node has to choose the MAC address from one of its interfaces, if it has more than one. This identity field has the same value for every SERVER_DISCOVERY and SERVER_POLL message emitted by the node, although it was doing from different interfaces.

1. *NIPs*: Amount of IP addresses solicited by the node. It will be equal to the interface number of client node (1 byte).
2. *Count*: Number of times that the SERVER_DISCOVERY petition has been tried (1 byte).

- *SERVER_OFFER*

Figure 4 shows the SERVER_OFFER message format.

**Figure 4.** SERVER_OFFER message format.

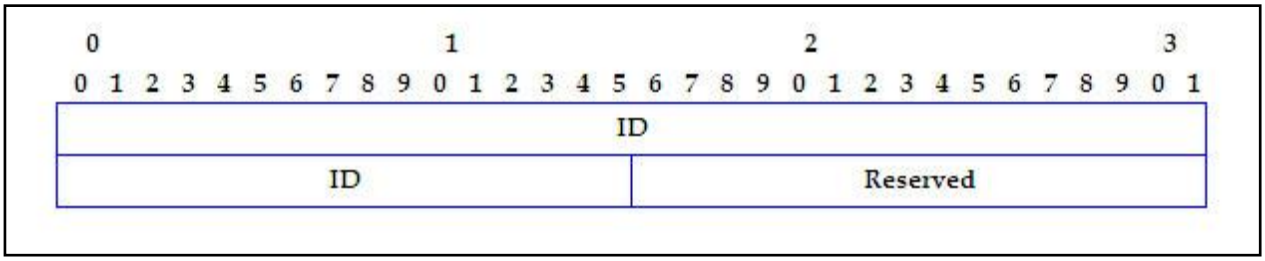



1. *Range*: Number of IP addresses offered (30 bits).
2. *Ready* (R): It indicates whether the node that offers the IP addresses is ready to assign them or only communicates their existence but at this point cannot assign them (1 bit).
3. *Local* (L): It indicates whether the range offered is from the sending node, or else be asked to turn to a third node (1 bit).

- *SERVER_POLL*

Figure 5 shows the SERVER_POLL message format.

**Figure 5.** SERVER_POLL message format.

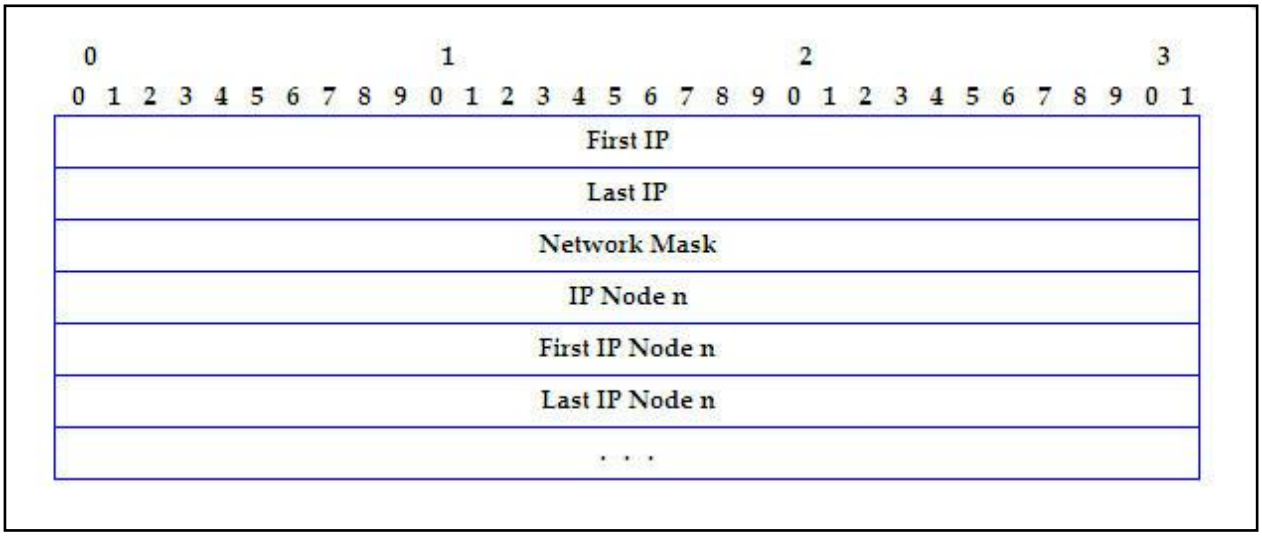



1. *ID*: Node identification (6 bytes). It is the same identification as that elected in the SERVER_DISCOVERY message.
2. *Reserved*: Reserved for future implementations (2 bytes).

- *IP_ASSIGNED*

Figure 6 shows the IP_ASSIGNED message format.

**Figure 6.** IP_ASSIGNED message format.

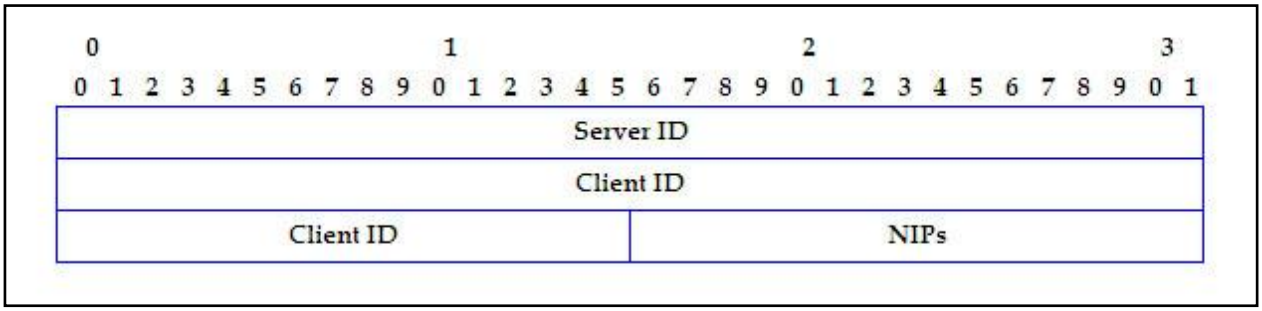



1. *First IP*: IP address start of a free address block (4 bytes).
2. *Last IP*: IP address end of free address block (4 bytes).
3. *Network Mask*: It consists of 4 bytes.
4. *IP Node n, First IP Node n, Last IP Node n*: This represents an entry on the free block table for all the nodes in the network. Each node is represented by these fields of 4 bytes, each one being: the node IP address, the initial IP address and the final from its free address block, respectively.

- *IP_RANGE_REQUEST*

  Figure 7 shows the IP_RANGE_REQUEST message format.

**Figure 7.** IP_RANGE_REQUEST message format.

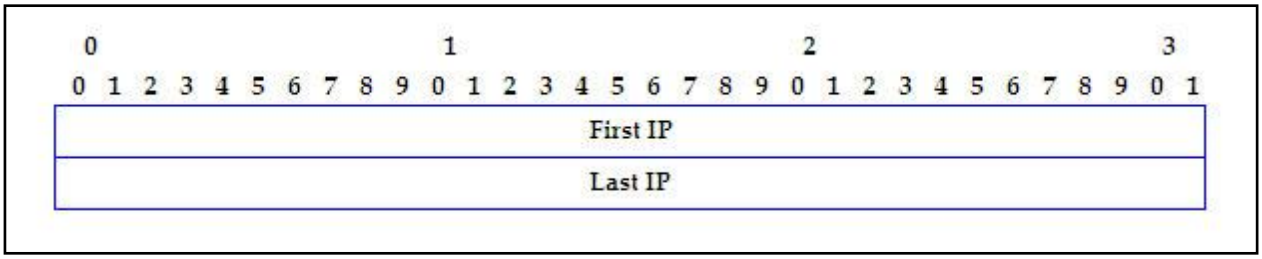



1. *Server IP*: Server address that requests the range for client (4 bytes). It can be different from the IP address of the sender message, by treating multi-hop networks.
2. *Client ID*: Client node identification. It has the same value as the ID field of the SERVER_DISCOVERY and SERVER_POLL message (6 bytes).
3. *NIPs*: Requested IP Address Number (2 bytes).

- *IP_RANGE_RETURN*

Figure 8 shows the IP_RANGE_RETURN message format.

**Figure 8.** IP_RANGE_RETURN message format.




1. *First IP*: The initial IP address of the free address block (4 bytes).
2. *Last IP*: The final IP address of the free address block (4 bytes).

### *3.5. Timers*

The wireless and mobile nature of MANET means that there are situations in the networks where messages are lost, or delayed in getting to its destination more than the estimated time. Therefore, we expose below a series of timers used to solve such situations:

- *SERVER_DISCOVERY_TIMER*: After sending the message, the client node will start this timer when it gets to the state WAITING_REPLY. During this time the node is waiting for SERVER_OFFER messaged of possible close by nodes belonging to a network.

The longer the timer runs, the more time it will dedicate to receiving messages of this kind, thus there will be more to process and, therefore, it is easier to get an address block. But it also means increasing the latency to obtain an IP address.

If this timer expires and the client node has not received any SERVER_OFFER, there has been a message loss, or perhaps there are no server nodes, it will send a new SERVER_DISCOVERY. This action is repeated a maximum number of times (SDISCOVERY_MAX_RETRY) and, if it goes on without receiving messages, it will initiate its own network.

- *SERVER_OFFER_TIMER*: After the SERVER_OFFER message, the node goes into the WAITING_POLL state, and starts this timer. When it expires, the state will change to IDLE.
- *SERVER_POLL_TIMER*: As soon as the SERVER_POLL message is sent, the node client will wait for the IP_ASSIGNED message for the time that this timer should determine. If this message does not come, the SERVER_POLL will be re-transmitted up to a maximum number of attempts, defined as SPOLL_MAX_RETRY. If the maximum number of attempts is exceeded, it will begin the configuration process again.
- *IP_RANGE_REQUEST_TIMER*: The server node who sends an IP_RANGE_REQUEST message to another node of the network initiates this timer at that moment. As with the SERVER_POLL_TIMER, if this timer expires, the IP_RANGE_REQUEST message will be forwarded up to a maximum number of attempts, given by RREQUEST _MAX_RETRY.

- *ACCEPTED_OFFER_TIMER*: This timer is activated after sending an IP_ASSIGNED message or an IP_RANGE_RETURN message. During this time, the server node cannot reply to requests of SERVER_POLL or IP_RANGE_REQUEST type. This restriction will arise after the timer expires (the offer expired without being accepted), or on having detected that a node with the first IP address of the offered ones has entered the network (the offered address block was accepted). It is necessary to bear in mind that although it could not assign IP address, the server node will keep on answering SERVER_DISCOVERY requests giving the value 0 to the field R (READY) in the SERVER_OFFER message. In this way, the node client is informed of the existence of the server, although it should not be capable of assigning IP address immediately.
- *NODE_DOWN_TIMER*: When OLSR erases the route towards a node, it is not eliminated immediately from the *Free_IP_Blocks* table. In its place, this timer is initiated. If before the timer expires it manages to discover a route to the node, that means that it disappeared momentarily, but it did not leave the network. Therefore, the elimination is cancelled on the *Free_IP_Blocks* table. In case the timer expires and a route has not been recovered, the node is assumed to be lost and its entry is eliminated from the *Free_IP_Blocks* table, updating those who match.
- *INIT_TABLE_TIMER*: On receiving the *Free_IP_Blocks* auto-configuration table in the IP_ASSIGNED message, the client node activates this timer. During that time, the table contains nodes so that OLSR does not have a well-known route yet. After the timer expires, the nodes from the *Free_IP_Blocks* table that do not have entries in the OLSR route table are verified: these nodes are eliminated (updating the corresponding entries), since they are nodes that belonged to the network on having received the table, and have left it before OLSR knew of its existence.
- *INIT_ASSIGN_TIMER*: This timer is used as much by the node client as by the server when they have received or assigned an IP address block, respectively. That is to say, the server initiates it, on having verified the node entry with the first IP address of the block offered in the IP_ASSIGNED message, and the client initiates it after receiving the IP_ASSIGNED message and to configure its address. Thus there has been time for the whole network to update its Free_IP_Blocks table before more changes take place. During that time, they will ignore SERVER_POLL or IP_RANGE_REQUEST messages, although they will reply to the SERVER_DISCOVERY.
- *NODE_DOWN_ASSIGN_TIMER*: When an already configured node detects the departure of another one, and it verifies that it is its turn to gather the IP address that remains free, it starts its timer. More concretely, the timer will be activated when the elimination of the OLSR routing table is detected, that is to say, it will be activated at the same time as the NODE_DOWN_TIMER timer.

Until it does not expire, the node will ignore the SERVER_POLL and IP_RANGE_REQ requests. This way, a margin of time will happen to ensure that all the nodes in the network detect the mentioned departure and update their *Free_IP_Blocks* table, before assigning them to some another new node. Therefore, the duration of this timer must be greater than that of the NODE_DOWN_TIMER to make sure that the rest of nodes in the network not only detect the elimination of a route but they have

eliminated it from the *Free_IP_Blocks* table. The node will keep on replying to the SERVER_DISCOVERY messages fixing the value 0 in the R field from the SERVER_OFFER message.

- *SLEEP_TIMER*: Timer used by a client node when it detects nearby nodes belonging to some network, but that are not in a position to assign IP addresses at this moment. This way, it gives a margin of time to allow the processes to end preventing them from assigning address.

### *3.6. State Diagrams*

Depending on whether they are in the process of joining the network, or if they already belong to one, two types of nodes are differentiated: client and server. In the following paragraphs the state diagrams that govern the behavior of both types of nodes are shown and explained. The state, in which the node is found, changes when sending or receptions of messages ocurrs, or when determined timers expire.

#### 3.6.1. Server Node

We call all the nodes in the network that are configured correctly server nodes, that is to say, they possess a valid IP address with which they can communicate with the rest of nodes, and a free IP address block. With this free address block they facilitate access to the new nodes, which we will call clients. Figure 9 shows the state diagram. As we can see, two types of states exist: the ones represented with rounded and clear rectangles, and those enclosed in somewhat darker rectangles with corners. We will call them states of the type *ready* or *not_ready*, respectively.

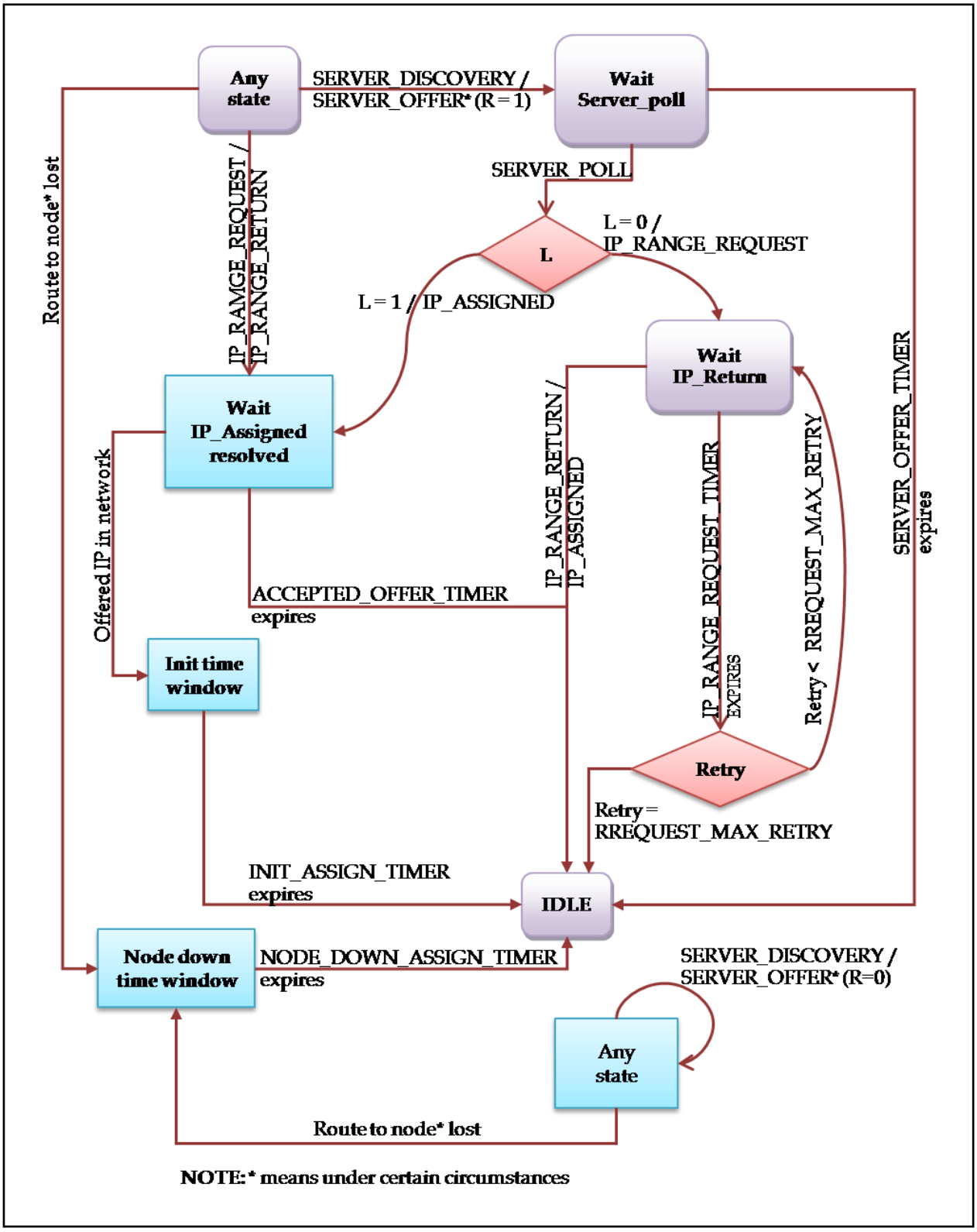
From any state, the server node is in expectation of SERVER_DISCOVERY messages. It will always reply with a SERVER_OFFER message, but depending on whether the current state of the node is of *ready* or *not_ready* type, it will answer giving to the field R (READY) the value 1 or 0. This way, a node always announces its presence, although at this precise moment it should not be capable of assigning IP addresses to a client. This is indicated in the state diagram with the any state state.

- *Any state (ready):* From any state of ready type, the server node will respond to a SERVER_DISCOVERY message with one of SERVER_OFFER type (R field with value 1). That will ensure that the server goes to the state Wait SERVER_POLL. A node can reply SERVER_DISCOVERY at the same time to requests from different nodes, and be awaiting any of the corresponding SERVER_POLL messages. Also they will answer messages of type IP_RANGE_REQUEST with one of the type IP_RANGE_RETURN. This means that whether the node is in any ready state, it will be give half of its free address blocks to any other node in the network without proper addresses and that it needs to facilitate the joining to a client.
- *Any state (not_ready):* Whilst the node is in a *not_ready* state, it will reply to the SERVER_DISCOVERY messages with a SERVER_OFFER message giving the value 0 to the field R.
- *Wait SERVER_POLL:* In this state the node waits for a time determined by the SERVER_OFFER_TIMER timer to receive a SERVER_POLL message. This message indicates that the client, the one who sent the SERVER_OFFER has chosen as its server for the process of auto-configuration. After the reception of the message, the node will send a

IP_ASSIGNED message to the client in the case of having available addresses locally (the field L of the message SERVER_OFFER had value 1); and it will pass to the Wait IP_Assigned state resolve. If the addresses that it offered were not local, it will have to ask for them from a node in the network with the IP_RANGE_REQUEST message; and it will pass into the state Wait IP_RReturn. In case any SERVER_OFFER messages are not received before the timer expires, the node will pass to be in the IDLE state.

- *Wait IP_ASSIGNED resolved:* In this state, of type not_ready, the node is waiting to find out if the client node correctly received the address block offered by an IP_ASSIGNED message, or through an IP_RANGE_RETURN message and an intermediary. The result can be that the client has been configured, or that it has not received the address block. The above-mentioned can take place for several reasons, since they can be problems with interferences in the message reception, movement of the client node out of the coverage range, and so on. If a new node appears in the network using the first IP address from the block offered in the message IP_ASSIGNED or IP_RANGE_RETURN, it means that the client node stopped being configured. In this moment the server node changes its state *to Init steal window*. If the node was not capable of finishing the auto-configuration process, the ACCEPTED_OFFER_TIMER timer will expire. In this case the node passes to the IDLE state.
- *Init time window:* This state serves to give a time margin that allows all the nodes in the network to be capable of detecting the joining of the recently configured client, before dividing again their own IP free address blocks. If this margin did not exist, and new requests would be attended immediately, synchronization problems might happen if other nodes were detecting the new incorporations to the network in an incorrect order.
- *Wait IP_return:* In this state the node is waiting for the reception of an IP_RANGE_RETURN message. When it receives this message, in which a node in the network indicates it has a block that it can offer the client in waiting; it will send an IP_ASSIGNED message to the client. After this, the node will have ended its function as server, and will pass to the IDLE state. If the IP_RANGE_RETURN message is not received before the timer IP_RANGE_REQUEST_TIMER expires, it will turn to try the request sending again an IP_RANGE_REQUEST message a maximum number of times RREQUEST_MAX_RETRY. These successive attempts are sent in every occasion to a different node. If the limit of attempts is exceeded, then the node will desist and change its state to IDLE.
- *IDLE:* This is the state of rest, or the one in which the node is idle. When it is in this state, the node does not undergo any operation related to the auto-configuration process. It is therefore treated as a waiting state.
- *Node down time window:* This node provides a time margin when the node must gather the IP address from a node that has left the network. More precisely, the node changes to this state on having detected that it has lost the route towards a node of whose address block it is responsible. This transition is done from any other state, be it of type ready or not_ready. After the time determined by the timer NODE_DOWN_ASSIGN_TIMER, the node will return to the state of rest IDLE. This timer is not to be confused with the NODE_DOWN_TIMER. Although they begin at the same time on having detected the same event, the processes involved are independent.

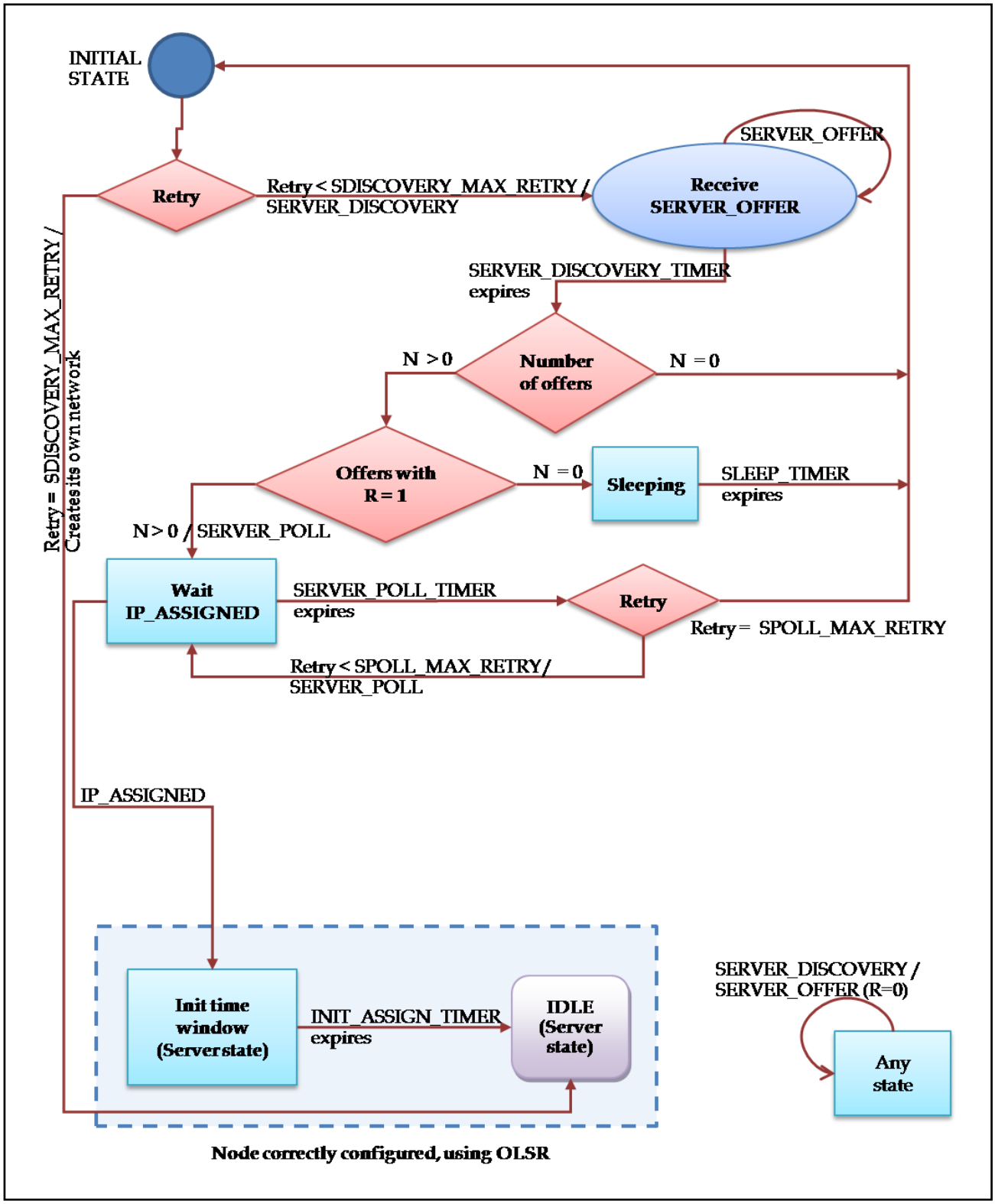
**Figure 9.** Server Node State Diagram.




3.6.2. Client Node

The procedure that a node that wants to gain access to a network follows is described in Figure 10.

**Figure 10.** Client Node State Diagram.




The diagram is provided with *not_ready* type states, explained in the state diagram of the server node and represented with the same style of rectangles. As if it were an already configured node, the node that is in process of configuration of its IP address replies to SERVER_DISCOVERY requests with SERVER_OFFER messages. In these messages the value 0 is given to the R field, since the node is not in a position to assign IP addresses, and only tries to announce its presence.

- *Initial state:* in which the auto-configuration process begins. If the number of attempts is less than the maximum, SDISCOVERY_MAX_RETRY, then the node emits a SERVER_DISCOVERY message for each of its network interfaces which are going to use the MANET network. It changes its state to Receive SERVER_OFFER. If it has gone over at the limit of attempts, then the node desists from his intention from finding a network to join and creates a new one. It will become a node server, beginning in the IDLE state of the server state diagram.
- *Receive SERVER_OFFER:* It is treated as a state of waiting, during which the SERVER_OFFER messages of other possible nodes are gathered. On having ended the waiting period, determined by the timer SERVER_DISCOVERY_TIMER, the responses are processed. If no SERVER_OFFER response has been received, it is returned to the initial state. If there is some offers, the number of them with the value 1 in the field R is verified. If among the offers none had the bit R set to 1, it means that there are nearby nodes belonging to a network, but at the moment they are not capable of assigning IP addresses. Therefore, the node passes to the Sleeping state.

If there was some offer with the bit R set to 1, the servers are sorted by preference and a SERVER_POLL message is sent to the first one of them. In this case the node changes its state to Wait IP_ASSIGNED. This state is not one of the types explained in the state diagram of a client node. This means that it does not reply to SERVER_DISCOVERY messages, since at the moment it does not know if there is any network nearby which to join.

- *Sleeping:* The node interrupts its attempts to join the network during the time determined by the SLEEP_TIMER timer. This is like that because there have been received SERVER_OFFER messages of nearby nodes that at the moment are not capable of assigning IP addresses, and what is claimed that after this time they are already capable of facilitating the join to the network. After the timer expires, the node will return to the *initial state.*
- *Wait IP_ASSIGNED:* In this state, the client is in expectation of an IP_ASSIGNED message on the part of the server to whom the message SERVER_POLL was sent. If the awaited message does not come, the SERVER_POLL_TIMER timer expires. In this case, it will turn to try to send a SERVER_POLL to the following server of the list generated after the end of the SERVER_DISCOVERY_TIMER timer. If the list of servers is ended, or it goes over the limit of attempts SPOLL_MAX_RETRY, the node returns to the initial state. On having received the IP_ASSIGNED message, the node configures its address (or addresses, in case of having several network interfaces). At this moment, it already takes part normally in the network, and passes to be a node server. The state with the one begins its behaviour as server is the *Init time window.*

## 4. Simulations and Results

Since in these networks nodes numbers that form the network are unpredictable, the protocol scalability is one of the main issues to consider. Therefore, it is essential to evaluate the impact of increasing the nodes number in the network in distinct parameters such as latency in address

assignment, the overhead because of control traffic or delay in the synchronization. Besides the nodes number in the network, it is necessary to take into account the frequency of the input and output nodes in the network. When a node leaves the network, the free address tables free in the network have been updated. If this is not done quickly, the network nodes cannot deal with requests for new entries in *the network to int*erpret that the network does not have free addresses.

To evaluate the D2HCP protocol performance has been used Network Simulator Network Simulator 3 (NS-3) [6]. Different scenarios of MANET networks were simulated to evaluate performance under different circumstances.

### *4.1. Simulation Scenarios*

Table 1 summarizes the main parameters used during simulations. When performing these simulations has been remained constant the entries number in the network per unit time. This factor is important, particularly in high density networks.

**Table 1.** Simulation Parameters.

| **Parameter** | **Value** |
|---|---|
| Simulation Area | 1,500 m × 1,500 m |
| Mobile Node Number | 50 to 1,600 |
| Mobility Pattern | Random Waypoint (*setdest*) |
| Routing Protocol | OLSR |
| Node Range or Coverage | 125 m |
| Simulation Number | 10 |
| Simulation Area | 1,500 m × 1,500 m |

### *4.2. Results*

Firstly the latency in the process of assigning addresses. Figures 11 and 12 show the evolution of the latency value depending on the nodes number in the network. Figure 11 shows the values using IPv4 addresses from class C, *i.e.*, with 254 available addresses. Figure 12 uses IPv4 addresses from Class B, providing 65534 addresses. Figure 11 shows that the increase in time address allocation begins to grow more quickly from the 125 nodes. However, using addresses from class B (Figure 12), the time experiences a very slight growth up to 1,600 nodes which were simulated.

These results indicate that the parameter that further determines the latency is the percentage of occupied addresses. When this percentage is nearly 50% the node number that do not have addresses to offer increases in direct proportion way. This does not allow a local assignment to be done and it is necessary to request the address from another node, increasing the time needed to complete the process. Anyway the average latency in the address assignment process is low.

Against auto-configuration protocols based on auto duplicate address detection (DAD), the protocol D2HCP also presents a great reduction in the overhead of control packets in the network.

In fact, in most cases, the configuration is performed locally, *i.e.*, a neighbor will assign address to the new node. This involves the sending of four control packets which do not spread to the rest of the network. In the case where no local address may be assigned, a *unicast* transmission is performed with

the chosen server, which causes much less overhead than a *broadcast* sending. The probability that cannot be assigned addresses locally depends on the relationship between the node number in the network and the available address number.

**Figure 11.** Latency in the IPv4 address assignment of a network from Class C.

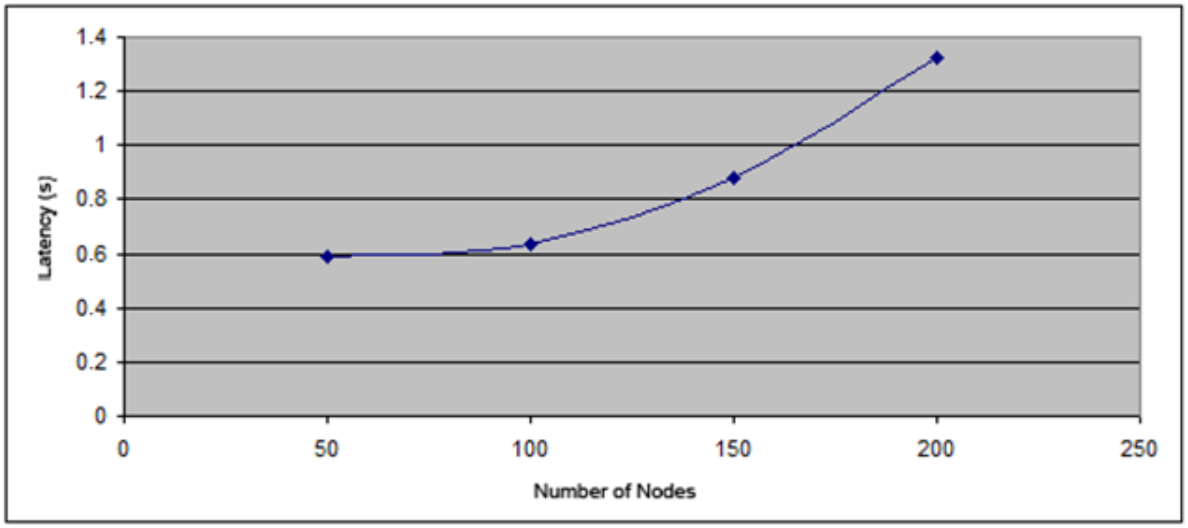
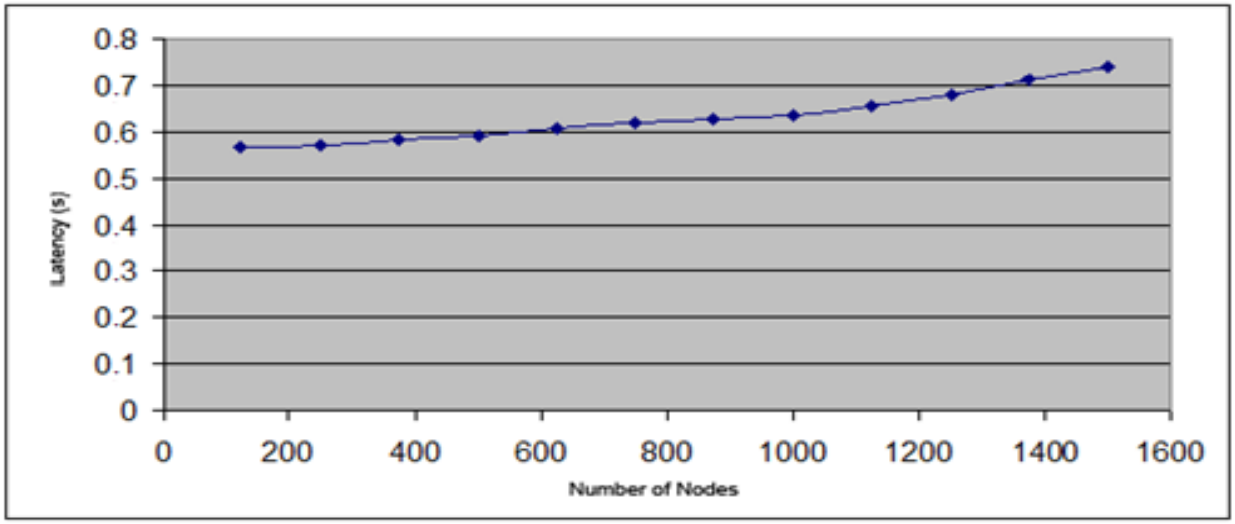



**Figure 12.** Latency in the IPv4 address assignment of a network from Class B.




In Figure 13 we can see the average number of control packets involved in each address configuration process. In the simulations have been used IP addresses from Class C, thus we have 254 network addresses. In Figure 13, you can see that when there are few nodes in the network, the required control message number to carry out auto-configuration is near the minimum, since in most cases the configuration can be performed locally.

However, when the free address number is close to 0, no configuration can be performed locally and remote nodes must use to perform such configuration, increasing the sent message number.

**Figure 13.** Control message average number sent in each address configuration.

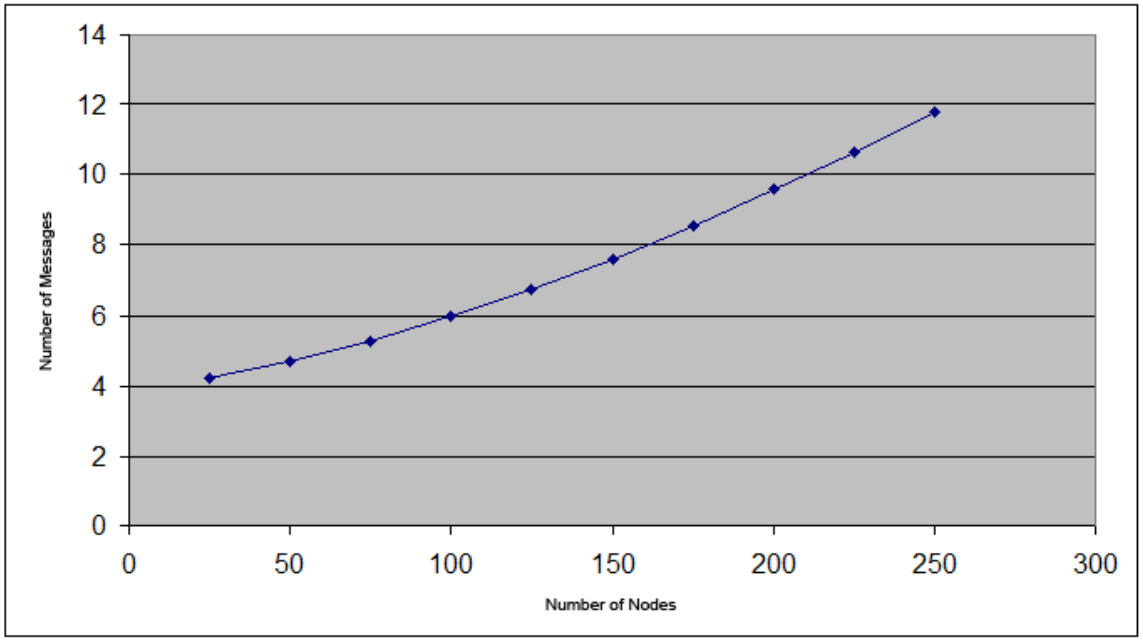
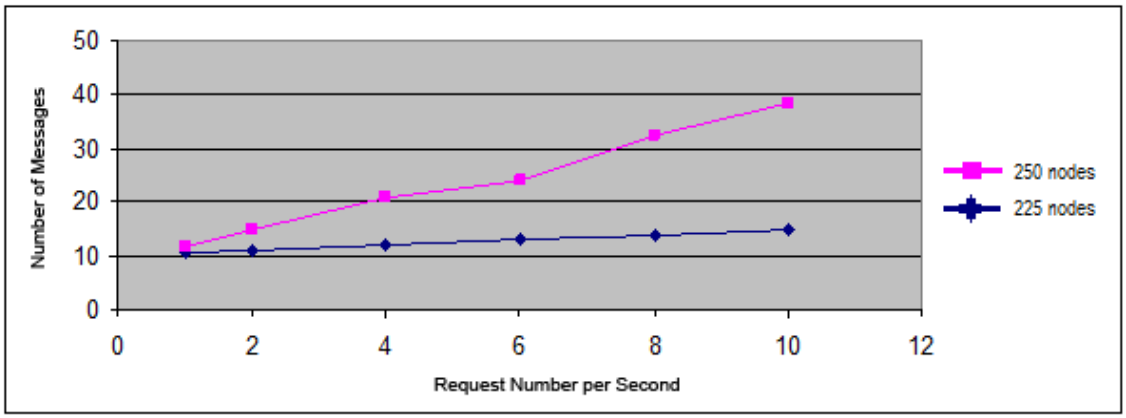



Figure 14 shows the evolution of the necessary control message number to assign an address based on the request number per second. In the case of the pink line the simulation was carried out in a scenario with 250 nodes. In the case of the blue line a scenario with similar characteristics has been used, but with 225 nodes. The used frequency of node departures in the network is similar to the frequency of node joins to hold the address availability.

As shown in Figure 14 in the case of a network with 225 nodes (about 90% occupancy) the required control messages number is practically independent from the request number per second, which means that the protocol efficiently supports the network scalability.

Only in the limit (around 100% occupancy) the protocol reduces its performance in terms of overhead. In fact, the main performance problem found is given in the situation that reflects the pink line in Figure 14. In situations where errors occur in the choice of the remote server, the latency increases in direct proportion to the control message number sent. However, this increase in overhead is acceptable, since it is still lower than the overhead incurred by the DAD algorithms.

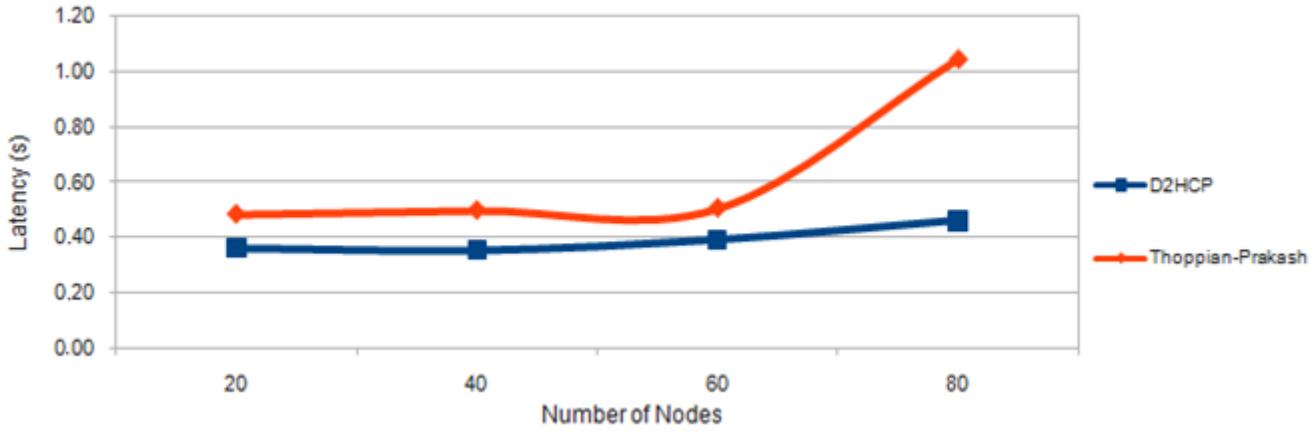
**Figure 14.** Control message number against to number of requests per second.



### *4.3. D2HCP* versus *the Thoppian and Prakash Protocol*

Figure 15 shows a comparison of the latency of D2HCP *versus* the Thoppian and Prakash Protocol. It is noted that in the first case the latency is lower than the second and it is very regular too (in Thoppian and Prakash the latency grows exponentially when the number of nodes is high) allowing us to conclude that D2HCP improves the results of its predecessor.

**Figure 15.** Latency D2HCP *versus* Thoppian and Prakash Protocol.




## 5. Conclusions and Future Work

An auto-configuration protocol for Mobile *Ad Hoc* Networks called D2HCP *Distributed Dynamic Host Configuration Protocol* (D2HCP) has been designed. This protocol is classified as a *stateful* protocol. This is an IPv4 address auto-configuration protocol for isolated Mobile *Ad Hoc* Networks.

In this protocol each node is responsible for managing a range of addresses. When a new node wants to begin participating in the network, one of the nodes within the network gives half of its address range to the new node. In the case of any adjacent node not having free addresses, but free addresses do exists, a request to a network node that has free addresses is done. In this operation mode is based on distributed nature of the protocol.

To keep updated information about free addresses owned by each node, the traffic of control packets from OLSR protocol. Such protocol at each node tries to keep updated knowledge of the whole topology from the network. This protocol has been designed to work together with OLSR; although it could operate with any proactive protocol by the flexibility of its design.

D2HCP warrants uniqueness for IP addresses in a wide variety of network conditions including message loss, concurrent requests and network partition. The simulation results show that the protocol has low latency and overhead. Worth noting is the protocol scalability features compared to other proposals in the literature, its flexibility that facilitates the protocol extension with new features, as well as synchronization process introduces null overhead.

Possible future work can be identified as follows:

- Detection of the *merging* to allow reassigning addresses that enters in conflict (something relatively easy since it would introduce a new message).

- Extension of the protocol to subordinate networks with access to the Internet or other networks, for which it should take into account the network topology to perform address auto-configuration process.
- Study protocol performance in cooperation with other proactive routing protocols (the first version D2HCP is designed to work together with OLSR).
- Add a security module that protects against different attackers to proportionate a safe auto-configuration.

## Acknowledgements

This work was supported by the Ministerio de Industria, Turismo y Comercio (MITyC, Spain) through the Project AVANZA I+D+I COMPETITIVIDAD TSI-020100-2010-482 and by the Ministerio de Ciencia e Innovación (MICINN, Spain) through the Projects TEC2007-67129/TCM and TEC2010-18894/TCM. This work was also supported by the Departamento Administrativo de Ciencia, Tecnología e Innovación (COLCIENCIAS, Colombia) through the Programa de Recuperación Contingente which funds the Project 121545221101. Ana Lucila Sandoval Orozco is also supported by the Programme Alβan, the European Union Programme of High Level Scholarships for Latin America, scholarship No. E07M403236CO.

## References

1. *Ad-Hoc Network Autoconfiguration Work Group*. Available online: http://tools.ietf.org/wg/autoconf/ (accessed on 25 November 2010).
2. Perkins, C.E.; Malinen, J.T.; Wakikawa, R.; Belding-Royer, E.M.; Sun, Y. *IP Address Autoconfiguration for Ad Hoc Networks*; Internet Draft; November 2001. Available online: http://tools.ietf.org/html/draft-perkins-manet-autoconf-01 (accessed on 25 November 2010).
3. Weniger, K. PACMAN: Passive Autoconfiguration for Mobile *Ad Hoc* Networks. *IEEE J. Sel. Area. Comm.* **2005**, *23*, 507–519.
4. Weniger, K. Passive Duplicate Address Detection in Mobile *Ad Hoc* Networks. In *Proceedings of the IEEE WCNC 2003*, New Orleans, LA, USA, March 2003. Available online: http://www.tm.uka.de/doc/2003/passive_dad_lsr_wcnc03.pdf (accessed on 25 November 2010).
5. Droms, R.; Bound, J.; Volz, B.; Lemon, T.; Perkins, C.; Carney, M. *Dynamic Host Configuration Protocol for IPv6 (DHCPv6)*; RFC 3315; July 2003. Available online: http://www.ietf.org/rfc/rfc3315.txt (accessed on 25 November 2010).
6. The NS-3 Network Simulator. Available online: http://www.nsnam.org/ (accessed on 25 November 2010).
7. Narten, T.; Nordmark, E.; Simpson, W.; Soliman, H. *Neighbor Discovery for IP version 6 (IPv6)*; RFC 4861; September 2007. Available online: http://www.ietf.org/rfc/rfc4861.txt (accessed on 25 November 2010).
8. Thomson, S.; Narten, T.; Jinmei, T. *IPv6 Stateless Address Autoconfiguration*; RFC 4862; September 2007. Available online: http://www.ietf.org/rfc/rfc4862.txt (accessed on 25 November 2010).

9. Cheshire, S.; Aboba, B.; Guttman, E. *Dynamic Configuration of IPv4 Link-Local Addresses*; RFC 3927; May 2005. Available online: http://www.ietf.org/rfc/rfc3927 (accessed on 25 November 2010).
10. Fazio, M.; Villari, M.; Puliafito, A. IP Address Autoconfiguration in *Ad Hoc* Networks: Design, Implementation and Measurements. *Comput. Netw.* **2005**, *50*, 898–920.
11. Li, L.; Cai, Y.; Xu, X.; Li, Y. Agent-based Passive Autoconfiguration for Large Scale MANETs. *Wirel. Pers. Commun.* **2007**, *43*, 1741–1749.
12. Kim, N.; Ahn, S.; Lee, Y. AROD: An Address Autoconfiguration with Address Reservation and Optimistic Duplicated Address Detection for Mobile *Ad Hoc* Networks. *Comput. Commun.* **2007**, *30*, 1913–1925.
13. Nesargi, S.; Prakash, R. *DADHCP*: *Distributed Dynamic Configuration of Hosts in a Mobile Ad Hoc Network*; Technical Report UTDCS-04-01; Department of Computer Science, University of Texas at Dallas: Dallas, TX, USA, January 2001.
14. Nesargi, S.; Prakash, R. MANETconf: Configuration of Hosts in a Mobile *Ad Hoc* Network. In *Proceedings of the IEEE INFOCOM* 2002, New York, NY, USA, June 2002; pp. 1059–1068. Available online: http://www.utdallas.edu/~ravip/papers/infocom2002.pdf (accessed on 25 November 2010).
15. Ros, F.; Ruiz, P.; Perkins, C.E. *Extensible MANET Auto-configuration Protocol (EMAP)*; Internet Draft; March 2006. Available online: http://tools.ietf.org/html/draft-ros-autoconf-emap-02 (accessed on 25 November 2010).
16. Sheu, J.P.; Tu, S.C.; Chan, L.H. A Distributed IP Address Assignment Scheme in *Ad Hoc* Networks. *Int. J. Ad Hoc Ubiquitous Comput.* **2008**, *3*, 10–20.
17. Hsu, Y.Y.; Tseng, C.C. Prime DHCP: A Prime Numbering Address Allocation Mechanism for MANETs. *IEEE Commun. Lett.* **2005**, *9*, 712–714.
18. Kim, S.; Lee, J.; Yeom, I. Modeling and Performance Analysis of Address Allocation Schemes for Mobile *Ad Hoc* Networks. *IEEE Trans. Veh. Technol.* **2008**, *57*, 490–501.
19. Chu, X.; Sun, K.; Sakander, Z.; Liu, J. Quadratic Residue Based Address Allocation for Mobile *Ad Hoc* Networks. In *Proceedings of the IEEE International Conference on Communications (IEEE ICC* 2008*)*, Beijing, China, May 2008, pp. 2343–2347.
20. McAuley, A.J.; Manousakis, K. Self-Configuring Networks. In *Proceedings of the Military Communications Conference (MILCOM)*, Los Angeles, CA, USA, October 2002.
21. Misra, A.; Das, S.; McAuley, A. Autoconfiguration, Registration and Mobility Management for Pervasive Computing. *IEEE Personal Commun.* **2001**, *8*, 24–31.
22. Bernardos, C.; Calderon, M.; Moustafa, H. *Ad-Hoc IP Autoconfiguration Solution Space Analysis*; Internet Draft; November 2008. Available online: http://tools.ietf.org/pdf/draft-bernardos-autoconf-solution-space-02.pdf (accessed on 25 November 2010).
23. Bernardos, C.; Calderon, M.; Moustafa, H. *Survey of IP Address Autoconfiguration Mechanisms for MANETs*; Internet Draft; November 2008. Available online: http://tools.ietf.org/html/draft-bernardos-manet-autoconf-survey-04 (accessed on 25 November 2010).

24. Bernardos, C.; Calderon, M.; Moustafa, H. *Evaluation Considerations for IP Autoconfiguration Mechanisms in MANETs*; Internet Draft; November 2008. Available online: http://www.it.uc3m.es/cjbc/papers/draft-bernardos-autoconf-evaluation-considerations-03.txt (accessed on 25 November 2010).
25. García Villalba, L.J.; García Matesanz, J.; Sandoval Orozco, A.L.; Márquez Díaz, J.D. Auto-configuration Protocols in Mobile *Ad Hoc* Networks. *Sensors* **2011**, *11*, 3652–3666.
26. The Internet Engineering Task Force (IETF). Available online: http://www.ietf.org/ (accessed on 25 November 2010).
27. Patchipulusu, P. Dynamic Address Allocation Protocols for Mobile *Ad Hoc* Networks. Master's Thesis, Texas A&M University: Dallas, TX, USA, August 2001.
28. Mohsin, M.; Prakash, R. IP Address Assignment in a Mobile *Ad Hoc* Network. In *Proceedings of the Military Communications Conference (MILCOM)*, Anaheim, CA, USA, September 2002; Volume 2, pp. 856–861. Available online: http://www.utdallas.edu/~ravip/papers/milcom02.pdf (accessed on 25 November 2010).
29. Thoppian, M.R.; Prakash, R. A Distributed Protocol for Dynamic Address Assignment in Mobile *Ad Hoc* Networks. *IEEE Trans. Mob. Comput.* **2006**, *5*, 4–19.
30. Clausen, T.; Jacquet, P. *Optimized Link State Routing Protocol (OLSR)*; RFC 3626; October 2003. Available online: http://www.ietf.org/rfc/rfc3626 (accessed on 25 November 2010).

# An Improved Buddy System Auto-Configuration Protocol for Mobile Ad Hoc Networks

Julián García Matesanz[#1], Luis Javier García Villalba[*2], Ana Lucila Sandoval Orozco[*3] and José René Fuentes Cortez[*4]

[#]*Grupo de Análisis, Seguridad y Sistemas (GASS)*
*Sección Departamental de Sistemas Informáticos y Computación – LSI y CCIA –*
*Facultad de Ciencias Matemáticas, Despacho 310-F*
*Universidad Complutense de Madrid (UCM)*
*Plaza de Ciencias, 3*
*Ciudad Universitaria, 28040 Madrid, Spain*

[1] julian@sip.ucm.es

[*]*Grupo de Análisis, Seguridad y Sistemas (GASS)*
*Departamento de Ingeniería del Software e Inteligencia Artificial (DISIA)*
*Facultad de Informática, Despacho 431*
*Universidad Complutense de Madrid (UCM)*
*Calle Profesor José García Santesmases s/n,*
*Ciudad Universitaria, 28040 Madrid, Spain*

[2] javiergv@fdi.ucm.es

[3] asandoval@fdi.ucm.es

[4] jrfuente@fdi.ucm.es

***Abstract* - Mobile ad hoc networks (MANETs) are multihop wireless networks of mobile nodes without any fixed or preexisting infrastructure. The topology of these networks can change randomly due to unpredictable mobility of nodes and propagation characteristics. In most networks, including MANETs, each node needs a unique identifier to communicate. This work presents a distributed protocol for dynamic IP address assignment to nodes in MANETs. Nodes of a MANET synchronize from time to time to keep a record of IP address assignment in the entire network and detect any IP address leaks. The proposed stateful autoconfiguration scheme uses the OLSR proactive routing protocol for synchronization and guarantees unique IP address under a variety of network conditions including message losses and network partitioning. Simulation results show that the protocol incurs low latency and communication overhead for an IP address assignment.**

***Index Terms* — Mobile ad hoc network, MANETs, IP address assignment, autoconfiguration, dynamic host configuration, stateful protocol, synchronization, OLSR proactive routing protocol.**

## I. INTRODUCTION

*Mobile Ad hoc NETwork* (MANET) is a set of mobile nodes which communicate themselves through wireless links. In contrast with the conventional networks, a MANET does not need a previous infrastructure, since nodes rely on each other to operate themselves, forming what is called multi-hop communication. Such networks have several disadvantages that a conventional network does not present.

The topology of this kind of network may change quickly and in an unpredictable way. Moreover, variations in the capacity of nodes and links, frequent errors in the transmission and security lack could occur.

Finally, limited resources of the nodes it must be taken into account since normally an ad hoc network will be formed by devices fed by batteries.

To communicate with each other [1] the ad hoc nodes need to configure their interfaces with local addresses which are valid inside an ad hoc network. The ad hoc nodes may also need to globally set routing addresses to communicate with other devices

in Internet. From the perspective of the IP layer, an ad hoc network is presented as a multi-hop network of level 3 constituted by a link collection.

In an autonomous ad hoc mobile network the nodes can be uniquely identified across an IP address with the only premise that this address must be different than the one of any other node in the network.

The process of configuration is the set of steps through which a node obtains its IP address within the network. There are two mechanisms to set addresses: *Stateless* and *Stateful*.

The *Stateless* address configuration proposes its own node to be the one in changed of generating its IP address. The address is obtained from the concatenation of a network well-known prefix and the theoretically unique number inside the network generated by the node. This mechanism may require the inclusion of a module in responsible for verifying the uniqueness of the generated address called *Duplicate Address Detection* (DAD) [2-4].

On the other hand, the *Stateful* address configuration is based on using servers which control and assign addresses to all the nodes of the network. *Dynamic Host Configuration Protocol* (DHCP) [5] is an example of *Stateful* configuration. However, because of the multi-hop nature of the mobile ad hoc networks, this protocol cannot be applied directly.

This work proposes a *Stateful*-based auto-configuration protocol, which guarantees uniqueness of IP address under a wide variety of network conditions as message missing and network partitioning.

This work is structured in 5 sections; the first one is the present introduction. Section 2 shows the obligated references in the auto-configuration protocol scope of mobile ad hoc networks. Section 3 contains a brief specification so-called *Distributed Dynamic Host Configuration Protocol* (D2HCP), a proposal about IP address auto-configuration for Mobile Ad Hoc Network.

Section 4 presents the simulations about protocol D2HCP carried out in NS-3 [6]. Finally, Section 5 shows the main advantages of the developed new protocol as well as potential future extensions to the study.

## II. Related Works

The mobile ad hoc networks present special features which must bear in mind when an address configuration protocol is implemented.

Many solutions exist for conventional networks (e.g.: RFCs 3315 [5], 4861 [7], 4862 [8] and so on) but the mobile ad hoc networks were not taken into account in its design. It is necessary, since, to give support multi-hop, support to dynamic topologies and to the *merging* and *partitioning* of networks, events that are typical in the mobile ad hoc networks.

There are numerous works that carry out proposals for the address configuration in a mobile ad hoc network using the *Stateless* as *Stateful* mechanism. Without doubt, the most representative are [2, 9-21].

Bernardos et al [22-24] carry out a rigorous study of the problems of the auto-configuration in mobile ad hoc networks, presenting an itemized review of the more representative auto-configuration protocols.

A comprehensive review of the main auto-configuration protocols can be found in [25].

Perhaps *Internet Engineering Task Force* (IETF) [26] has the most well-known work group called *Ad-Hoc Network Autoconfiguration Work Group* (Autoconf WG) [1] which its principal purpose is to describe the addressing model for ad hoc networks and how the nodes set its addresses in these networks. It is essential that such models do not cause problems to other components of an ad hoc system such as standard applications which are executed in an ad hoc node or Internet nodes connected to the ad hoc nodes. The work of this group can include the development of new protocols whether the existing IP auto-configuration mechanisms turn out to be inadequate. Nevertheless, the first task of this work group is to describe a practical addressing model for ad hoc networks.

The solutions described previously have supposed significant contributions to aid our comprehension of the problem. Nevertheless, we think that all these approaches handle only a subset of the network conditions enumerated as follows:

1. Dynamic Topology: The nodes in the network move arbitrarily and can join and leave the network dynamically.
2. Message loss and failure in the nodes: message loss can be quite frequent and can duplicate the IP address allocation if it is not managed correctly. The nodes can abruptly depart from the network due to a link failure or an accident.
3. *Partitioning* and *merging*: The network can split into multiple networks and, later, join with others. During network merging it is possible to have duplicated IP addresses in the fused network.
4. Address concurrent requests: Multiple nodes may want to join the network simultaneously.
5. Limited Energy and Bandwidth: The nodes in a mobile ad hoc network have limited energy and the links have a limited wideband. Therefore, the communication overhead which is incurred should be low.

In this work a solution similar to DAAP [27, 28] and to [29] that guarantees uniqueness in the IP address allocation under a wide set of network conditions is proposed. In our approach, the majority of address allocations imply local communication causing low communication overhead and low latency.

## III. D2HCP (Distributed Dynamic Host Configuration Protocol)

The *Distributed Dynamic Host Configuration Protocol* (D2HCP) is an auto-configuration protocol that manages the joining and departing of nodes in MANET.

The protocol makes the MANET nodes collaborate with each other to manage the assignment of unique and correct IP addresses in a distributed manner. All the network nodes have the same role; there is no special type of node that centralizes the management of the same.

Nodes have a synchronization system is based on the OLSR [30] routing protocol. Thanks to this mechanism, the synchronization is done passively, monitoring the mentioned routing protocol, thus no overhead is generated in the network traffic compared to that one generated by the OLSR protocol.

Due to the fact that all the nodes are responsible for managing the joining of any new node to the network, this process can be done quickly. A node that wishes to join a network tries to contact any node still belonging to it, and may receive several responses from multiple nodes. This makes the chances of successfully joining the network high, because of the high availability and redundancy that the distributed management disposes.

Here we introduce the D2HCP specification: it begins with the used data structures, continues with an explanation of the exchanged messages between nodes for join and departure of these is continued, and then details how synchronization takes place in protocol is detailed. Finally, we explain the exchanged messages format during the auto-configuration process, detailing how to solve the possible message loss in the network using appropriate timers and performing certain actions when they expire to restore the auto-configuration process, as well as state diagrams for each operation mode that can take a node.

### *A. Data Structures*

The data structures of this protocol can be classified into those handling the auto-configuration mechanism and those belonging to the OLSR routing protocol.

OLSR internally stores a routing table which is updated periodically. This table contains information about the route to each node, stored in the following fields:

- R_dest_addr: IP address of the destination node.
- R_next_addr: IP address of next hop in the route.
- R_dist: Distance to the destination node.
- R_iface_addr: IP address of the outgoing interface to the destination node.

The structures necessary for auto-configuration are:

- IP addresses of the node interfaces.
- Netmask.
- Free_IP_Blocks: A table of free block from each node in the network.

### *B. Joining and Departing of Nodes*

*1) Node Join*: The protocol uses a specific message number for each operation. All the operations are defined looking for optimum working and low latency.

This section discusses how communication is established between the nodes and the messages transmitted during the joining and departure of nodes in the network.

The entry of a node to the network implies the need to find a node acting as a server. Once found, it will facilitate the joining by providing an IP address block and a table Free_IP_Blocks representing the state of all the nodes in the network.

Until the node does not have an assigned IP address, its communication with nodes which might act as servers will be through the MAC layer.

The configuration mechanism uses 4 types of messages in most cases. If no nodes in range with free IP addresses will be used 6 types of messages in total.

1. *SERVER_DISCOVERY*: The client node wishing to join a network starts the process with a message of this type. It is transmitted by the MAC layer, with the *broadcast* address as its destination. The message indicates the IP address number which is required (equal to the interface number).

   If the node has more than one network interface, the message is transmitted through all of them, using the ID field thus the different interfaces are not confused with several nodes.

2. *SERVER_OFFER*: The network nodes receiving the message *SERVER_DISCOVERY* reply to this message, also using the MAC layer, in which an IP address number is offered. The number of addresses offered is half of the available range.

   The *SERVER_DISCOVERY* message includes a field Count indicating how many attempts have been made by the client. Depending on its value, the server nodes will behave as follows:

   * *Count* = 1: The server node will respond with a *SERVER_OFFER* if enough addresses are available and the fields R (*Ready*) and L (*Local*) will have the value 1 (it can assign the addresses provided at the moment, and they are addresses of block of own node).
   * *Count* = 2: The node server will respond with a *SERVER_OFFER* if the fields can take the value R = 1 and L = 1. If not possible, it will still also respond if it is the case that there are enough addresses and R = 0, L = 1 (the server can not assign addresses at the moment, but it has them).

- *Count* > 2: If the node has addresses available and is in a capable state to do so, it will send a *SERVER_OFFER* with R = 1, L = 1. If it can, will send it with R = 0, L = 1. And finally, if it does not have enough free addresses, it will send the message with the fields R = 1, L = 0 (immediate availability of addresses, but the offered addresses are from another node in the network).

3. *SERVER_POLL*: After a time of hearing, the client node will have received several messages *SERVER_OFFER*. If not, it will try again.

It will sort received messages by the following criteria:

- The servers which are not available are discarded, i.e. with R = 0. The SERVER_OFFER with R = 0 is not used to reply with a SERVER_POLL, but they have the function of informing the client that there is a server node in the network although it cannot provide access to it at this moment.
- Priority to local addresses is given: it will prefer messages with the field L = 1.
- Finally, it is organized so that the offered address number are ranked, from highest to lowest.

According to this criteria for order preference, it will send a message *SERVER_POLL* to the first server (by the MAC layer, again) to let it know that the node has chosen this one to assign a free IP address block to it.

4. *IP_RANGE_REQUEST*. If the addresses provided by the server node were not their own, but they were from a third node in the network, with this message there will be a formal request made to that node. Since there is communication between two nodes already configured correctly, it is performed at the IP layer.
5. *IP_RANGE_RETURN*. The third network node authorizes the node that sends the message *IP_RANGE_REQUEST* to assign the address block indicated in this message to client nodes. It is also a message sent by IP.
6. *IP_ASSIGNED*. After receiving the *SERVER_POLL*, if the provided addresses were of the own server node, or after *IP_RANGE_RETURN* message if it has been necessary to request the address from a third node, the node server sends this message to the client. This message is transmitted by the MAC layer. In this message the free address block which is assigned to the client and *Free_IP_Blocks* table representing the network state is indicated. The table which is transmitted in this message does not reflect the joining of the client node.

After this message exchange, the client node chooses the first one of the block which has been assigned as its IP address. In the case of having more than one network interface, it will use the first ones of block in order, and will be the first of all which use as the primary address that identify the node.

*2) Node Departure*

The node departure mechanism does not require the exchange of any message. The node that wants to leave the network does not have to notify any other node of its departure, avoiding the overhead that these messages occur.

The other nodes in the network will become aware of the departure node through periodic updates of routes that OLSR protocol performs every so often. They will note that it has lost the path to that node, and therefore they removed it from its Free_IP_Blocks table, adding its free address block to the corresponding node as explained in the previous section.

### *C. Synchronization*

The synchronization is done by monitoring the routing table of routing protocol OLSR [30]. The joining or departure of a node in the network is detected when OLSR adds a new route to its routing table, or deletes an existing one. By detecting the joining or departure of a node in the network, it is updated locally, and without exchanging any message, Free_IP_Blocks table.

For this reason, the following rules are obeyed:

- The responsibility of recovering the IP addresses that a node which leaving the network makes available is one that can be attached to the right of the free block. This will not be possible when the block to be collected contains the lowest address of the network. In that case, the node that picks the block up is one that can add to it by the left.
- By dividing the free addresses in two blocks to deliver one of them to a new node that joins the network, the node that acts as server delivers the sub-block, which does not contain its own IP address, to the client.

When the node departure is detected, its entry must be removed, and an update of the corresponding nodes, now available IP address must be recorded.

By detecting the joining of a new node, a new entry in the table for it is created, and the free address block of the node which supplied its IP address will be updated. To know who this node which acted as server was, it is simply necessary to find out which node has the IP address of the new node in its free address block.

## IV. Simulations and Results

In the last network information discovery technique, the mobile node can consult a MIIS server, which stores information from several access networks and operators. To access this information, the MN must perform some steps before obtaining the desired information. It may require link-layer supports, transport protocol capability and security considerations. The main advantage of using such a technique is that the MN may have a complete and consistent view of the whole network. In addition, this approach allows MN mobility over several networks and operators. In this work, we will use this approach to demonstrate the benefits of use MIIS server technique in a heterogeneous mobile network environment.

### *A. Simulation Scenarios*

Table 1 summarizes the main parameters used during simulations.

When performing these simulations has been remained constant the entries number in the network per unit time. This factor is important, particularly in high density networks.

**Table I.**
Simulation Parameters.

| Parameter | Value |
|---|---|
| Simulation Area | 1500 m x 1500 m |
| Mobile Node Number | 50 to 1600 |
| Mobility Pattern | Random Waypoint (*setdest*) |
| Routing Protocol | OLSR |
| Node Range or Coverage | 125 m |
| Simulation Number | 10 |
| Simulation Area | 1500 m x 1500 m |

### *B. D2HCP vs Thoppian-Prakash Protocol*

Figure 1 shows a comparison of the latency of D2HCP versus Thoppian and Prakash's Protocol. It is noted that in the first case the latency is lower than the second and it is very regular too (in Thoppian and Prakash the latency grows exponentially when the number of nodes is high) allowing us to conclude that D2HCP improves the results of its predecessor.

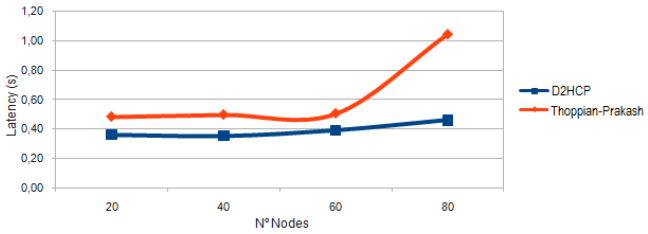



Fig. 1 D2HCP vs Thoppian-Prakash Protocol

## V. Conclusion

An auto-configuration protocol for mobile ad hoc networks called D2HCP Distributed Dynamic Host Configuration Protocol (D2HCP) has been designed. This protocol is classified as a stateful protocol. This is an IPv4 address auto-configuration protocol for isolated mobile ad hoc networks. Each node is responsible for managing a range of addresses. When a new node wants to begin participating in the network, one of the nodes within the network gives half of its address range to the new node. In the case of any adjacent node not having free addresses, but free addresses do exists, a request to a network node that has free addresses is done. In this operation mode is based on distributed nature of the protocol.

To keep updated information about free addresses owned by each node, the traffic of control packets from OLSR protocol. Such protocol at each node tries to keep updated knowledge of the whole topology from the network. This protocol has been designed to work together with OLSR; although it could operate with any proactive protocol by the flexibility of its design.

D2HCP warrants uniqueness for IP addresses in a wide variety of network conditions including message loss, concurrent requests and network partition. The simulation results show that the protocol has low latency and overhead. Worth noting is the protocol scalability features compared to other proposals in the literature, its flexibility that facilitates the protocol extension with new features, as well as synchronization process introduces null overhead. Possible future work can be identified as follows:

- Detection of the merging to allow reassigning addresses that enters in conflict.
- Extension of the protocol to subordinate networks with access to the Internet or other networks, for which it should take into account the network topology to perform address auto-configuration process.
- Study protocol performance in cooperation with other proactive routing protocols.
- Add a security module that protects against different attackers to proportionate a safe auto-configuration.

## ACKNOWLEDGMENTS

This work was supported by the Ministerio de Industria, Turismo y Comercio (MITyC, Spain) through the Project Avanza Competitividad I+D+I TSI-020100-2010-482 and the Ministerio de Ciencia e Innovación (MICINN, Spain) through the Project TEC2010-18894/TCM.

## REFERENCES

[1]. Ad-Hoc Network Autoconfiguration Work Group (autoconf). Available online: http://tools.ietf.org/wg/autoconf/ (accessed on 25 November 2010).

[2] Perkins, C.E.; Malinen, J.T.; Wakikawa, R.; Belding-Royer, E.M.; Sun, Y. *IP Address Autoconfiguration for Ad Hoc Networks*; Internet Draft; November 2001.

[3] Weniger, K. PACMAN: Passive Autoconfiguration for Mobile Ad hoc Networks. *IEEE J. Sel. Areas Commun.* **2005**, *23*, 507-519.

[4] Weniger, K. Passive Duplicate Address Detection in Mobile Ad Hoc Networks. In *Proceedings of IEEE WCNC 2003*, New Orleans, Louisiana, USA, March 2003.

[5] Droms, R.; Bound, J.; Volz, B.; Lemon, T.; Perkins, C.; Carney, M. *Dynamic Host Configuration Protocol for IPv6 (DHCPv6)*; RFC 3315; July 2003.

[6] The NS-3 network simulator. Available online: http://www.nsnam.org/ (accessed on 25 November 2010).

[7] Narten, T.; Nordmark, E.; Simpson, W.; Soliman, H. *Neighbor Discovery for IP version 6 (IPv6)*; RFC 4861; September 2007.

[8] Thomson, S.; Narten, T.; Jinmei, T. *IPv6 Stateless Address Autoconfiguration*; RFC 4862; September 2007.

[9] Cheshire, S.; Aboba, B.; Guttman, E. *Dynamic Configuration of IPv4 Link-Local Addresses*; RFC 3927; May 2005.

[10] Fazio, M.; Villari, M.; Puliafito, A. IP Address Autoconfiguration in Ad Hoc Networks: Design, Implementation and Measurements. *Comput. Netw.* **2005**, *50*, 898-920.

[11] Li, L.; Cai, Y.; Xu, X.; Li, Y. Agent-Based Passive Autoconfiguration for Large Scale MANETs. *Wirel. Personal Commun.* **2007**, *43*, 1741-1749.

[12] Kim, N.; Ahn, S.; Lee, Y. AROD: An Address Autoconfiguration with Address Reservation and Optimistic Duplicated Address Detection for Mobile Ad Hoc Networks. *Comput. Commun.* **2007**, *30*, 1913-1925.

[13] Nesargi, S.; Prakash, R. *DADHCP: Distributed Dynamic Configuration of Hosts in a Mobile Ad Hoc Network*; Technical Report UTDCS-04-01; University of Texas at Dallas, Department of Computer Science: Dallas, TX, USA, January 2001.

[14] Nesargi, S.; Prakash, R. MANETconf: Configuration of Hosts in a Mobile Ad Hoc Network. In *Proceedings of IEEE INFOCOM 2002*, New York, USA, June 2002; pp. 1059-1068.

[15] Ros, F.; Ruiz, P.; Perkins, C.E. *Extensible MANET Auto-configuration Protocol (EMAP)*; Internet Draft; March 2006.

[16] Sheu, J.P.; Tu, S.C.; Chan, L.H. A Distributed IP Address Assignment Scheme in Ad Hoc Networks. *Int. J. Ad Hoc Ubiquitous Comput*. **2008**, *3*, 10-20.

[17] Hsu, Y.Y.; Tseng, C.C. Prime DHCP: a Prime Numbering Address Allocation Mechanism for MANETs. *IEEE Communications Letters*. **2005**, 9, 712-714.

[18] Kim, S.; Lee, J.; Yeom, I. Modeling and Performance Analysis of Address Allocation Schemes for Mobile Ad Hoc Networks. *IEEE Transactions on Vehicular Technology*. **2008**, 57, 490-501.

[19] Chu, X.; Sun, K.; Sakander, Z.; Liu, J. Quadratic Residue Based Address Allocation for Mobile Ad Hoc Networks. In *Proceedings of IEEE International Conference on Communications (IEEE ICC 2008)*, Beijing, China, May 2008, pp. 2343-2347.

[20] McAuley, A.J.; Manousakis, K. Self-Configuring Networks. In *Proceedings of Military Communications Conference (MILCOM)*, Los Angeles, CA , USA, October 2002.

[21] Misra, A.; Das, S.; McAuley, A. Autoconfiguration, Registration and Mobility Management for Pervasive Computing. *IEEE Personal Communications*. **2001**, 8, 24-31.

[22] Bernardos, C.; Calderon, M.; Moustafa, H. *Ad-Hoc IP Autoconfiguration Solution Space Analysis*; Internet Draft; November 2008.

[23] Bernardos, C.; Calderon, M.; Moustafa, H. *Survey of IP Address Autoconfiguration Mechanisms for MANETs*; Internet Draft; November 2008.

[24] Bernardos, C.; Calderon, M.; Moustafa, H. *Evaluation Considerations for IP Autoconfiguration Mechanisms in MANETs*; Internet Draft; November 2008.

[25] García Villalba, L.J.; García Matesanz, J.; Sandoval Orozco, A.L.; Márquez Díaz, J.D. Auto-Configuration Protocols in Mobile Ad Hoc Networks. *Sensors*, **2011**, 11(4), 3652-3666.

[26] The Internet Engineering Task Force (IETF). Available online: http://www.ietf.org/ (accessed on 25 November 2010).

[27] Patchipulusu, P. *Dynamic Address Allocation Protocols for Mobile Ad Hoc Networks*. Master's Thesis, Texas A&M University: Dallas, TX, USA, August 2001.

[28] Mohsin, M.; Prakash, R. IP Address Assignment in a Mobile Ad Hoc Network. In *Proceedings of Military Communications Conference (MILCOM)*, Anaheim, California, USA, September 2002; Volume 2, pp. 856-861.

[29] Thoppian, M.R.; Prakash, R. A Distributed Protocol for Dynamic Address Assignment in Mobile Ad Hoc Networks. *IEEE Trans. Mob. Comput.* **2006**, *5*, 4-19.

[30] Clausen, T.; Jacquet, P. *Optimized Link State Routing Protocol (OLSR)*; RFC 3626; October 2003.

Published in IET Information Security
Received on 13th April 2010
Revised on 31st October 2010
doi: 10.1049/iet-ifs.2010.0085

ISSN 1751-8709

# Secure extension to the optimised link state routing protocol

*L.J. García Villalba*[1] *J. Garcia Matesanz*[2] *D. Rupérez Cañas*[1] *A.L. Sandoval Orozco*[1]

[1]*Departamento de Ingeniería del Software e Inteligencia Artificial, Grupo de Análisis, Seguridad y Sistemas (GASS), Universidad Complutense de Madrid*
[2]*Sección Departamental de Sistemas Informáticos y Computación, Grupo de Análisis, Seguridad y Sistemas (GASS), Universidad Complutense de Madrid*
*E-mail: javiergv@fdi.ucm.es*

**Abstract:** The design of routing protocols for mobile *ad hoc* networks rarely contemplates, in most cases, hostile environments. Consequently, it is common to add security extensions afterwards. One of the most important routing protocols is the optimised link state routing (OLSR), which in its specification assumes the trust of all nodes in the network, making it vulnerable to different kinds of attacks. This study presents an extension of OLSR, called COD-OLSR, which provides security for OLSR in the case of incorrect message generation attacks which can occur in two forms (identity spoofing and link spoofing). This is one of its main features, which takes into account the current topology of the node sending the message. The behaviour of COD-OLSR against different attackers in a variety of situations is evaluated. The simulation results show that COD-OLSR adds a slight overhead to OLSR and barely affects performance. The results also show that COD-OLSR is an interesting alternative to provide integrity in OLSR compared with classical mechanisms making use of cryptography, which is more complex and has a high overhead.

## 1 Introduction

Mobile *ad hoc* networks (MANETs) [1] are formed by mobile devices that can communicate with each other without appealing to a preexisting network infrastructure. Most routing protocols for these networks are designed without taking into account the possible malicious behaviour of any of the nodes, something that can be exploited to violate the security of the network. Optimised link state routing (OLSR) [2] belongs to this group of protocols where attackers can alter their behaviour, hence the need for an extension of this protocol. There are several techniques to provide integrity to the OLSR protocol. One of the most widespread is the use of digital signatures for authentication of the OLSR routing messages, of which there are two different variants or approaches: hop-by-hop and end-to-end.

Halfslund *et al.* [3] present a hop-by-hop approximation, where each node signs OLSR packets that are transmitted, discarding the signature of the previous node. This signature only verifies that the node which forwards the message is the same as the one that signs it, but it does not verify the authenticity of the original message. The implemented solutions use shared keys (symmetric) for signature creation and verification. However, they do not mention how the administration/exchange of keys or the initial authentication is carried out. They also propose a method to prevent replay attacks using timestamps, but the overhead increases significantly with this method.

Adjih *et al.* [4] propose schemes to authenticate OLSR messages in an end-to-end approach so that the nodes that receive OLSR messages can authenticate the node which generated the original message. However, replay attacks are still possible. To avoid such attacks another approach has been developed which is based on a distributed timestamp that can verify whether a message is obsolete or not.

Raffo *et al.* [5] design another end-to-end extension where the nodes send OLSR messages as well as an ADVanced Signature message (ADVSIG). Nodes that announce links sign the messages to authenticate the node that originated the message. Mechanisms are established which improve the protection of the protocol against external attacks and from other nodes, although this does significantly add to the overheads.

Raffo *et al.* [6] supply a method that includes the geographical position of the sender node in the control messages as well as an evaluation of the similarity of the links at the same time. This technique has the disadvantage of requiring specific hardware (GPS and directional antennas).

Papadimitratos and Haas [7] provide security in the routing of the link state by using asymmetric primitives. To do this, they assume that each node in the network has a pair of public/private keys, and diffusing in broadcast its public key certificate to the other nodes that are within $N$ hops. These broadcast messages can be periodic or not, depending on changes in the topology of the network, allowing a new node to find out the key to enter the area. Despite using distributed certification the overhead introduced is also important.

Finally, Nait-Abdesselam [8] presents a scheme for detecting and preventing 'Relay' attacks (Wormhole attacks). This technique is based on firstly an identification of the potential links in which the attack occurs, then in distinguishing the links where the Wormhole attack occurs from the ones that

are correct. In order to achieve this, encrypted packets are exchanged between the two neighbours that carried out the attack. This solution is independent of any time synchronisation or location of information, but it has the disadvantage of increasing the overhead because of the exchange of messages for attack detection.

This paper presents a framework that allows us to maintain the integrity of OLSR messages with minimal overhead. It is structured into five sections. In this first section the problem is introduced through displays of related works in which various security extensions to OLSR are discussed. In Section 2, we recall some features of OLSR and we briefly analyse the major attacks it has been exposed to. In Section 3, we specify the security extension, providing the main results of this work. Section 4 contains the most relevant results of the simulation. Finally, Section 5 contains conclusions as well as directions for future research.

### 1.1 Attacks on OLSR

As its name suggests, the OLSR protocol uses an optimised flooding mechanism to disseminate information to the nodes and to build the network topology. Each node selects a set of neighbour nodes as multipoint relays (MPRs) that are responsible for relaying the control traffic (see Fig. 1) that is diffused in the network. In order to effect this diffusion process, the control message HELLO and TC are used.

HELLO messages allow the neighbouring nodes to be known and can calculate the MPRs. HELLO messages are exchanged periodically only between neighbour nodes, providing several lists of neighbours for each node. One of the lists contains the neighbours that have been picked up by the current node, but have not confirmed the bidirectional link yet. Another list is made up of neighbours who have established a bidirectional communication (symmetrical link). The last list contains neighbour nodes that have been selected by the origin of the HELLO message to act as MPR. In Fig. 2, we show the format of the HELLO messages and their fields.

TC messages are periodically sent through the MPRs. Its purpose is to diffuse topology information to the entire network. A TC message contains the set of bidirectional links between a node and its neighbours. In Fig. 3, we show the TC message format. The list of neighbours is included in this collection of the MPR selector set (MS), that is, those nodes that have chosen to emit the TC message as MPR.

There are other control messages with different functionalities, but we will not consider them, since they are not the main goal of this research.

The main attacks [9, 10], which OLSR is exposed to, are closely related to previous control messages. We distinguish the following kinds of attacks:

- *Incorrect message generation (IMG)*: A node can have bad behaviour in the generation of messages in the following cases: (i) when there is identity spoofing, that is, a node is incorrectly identified as another one, and (ii) when a node announces links with incorrect information in the control messages (link spoofing).

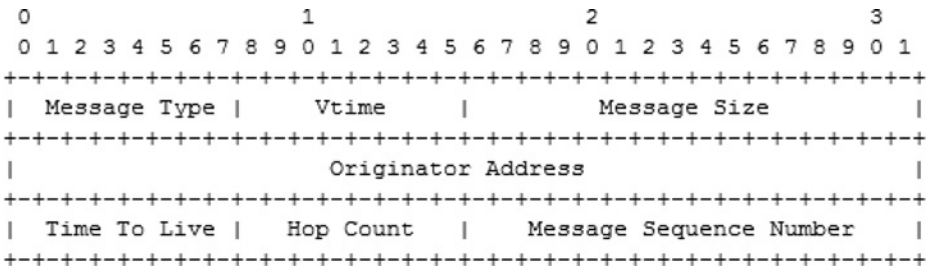



**Fig. 1** *OLSR message header format*

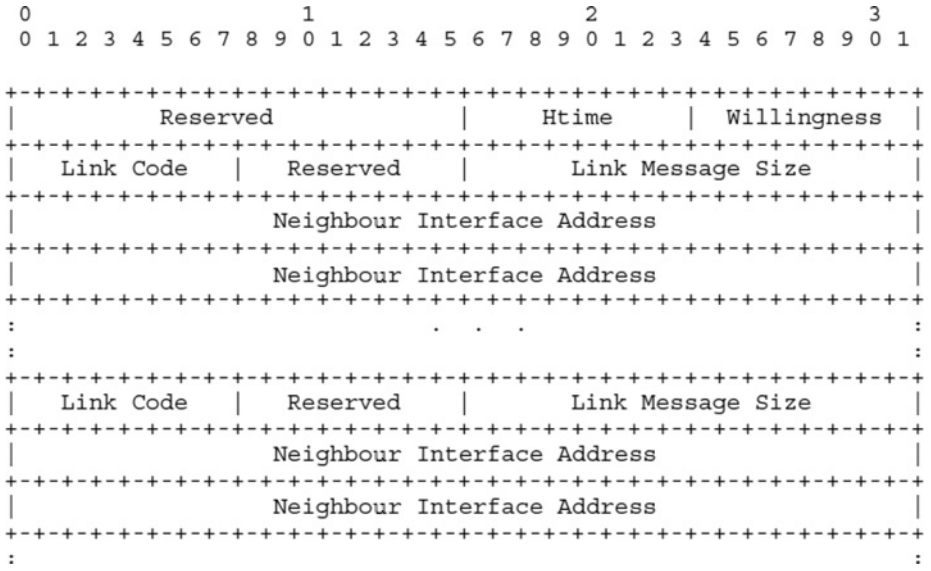



**Fig. 2** *HELLO message format*




doi: 10.1049/iet-ifs.2010.0085

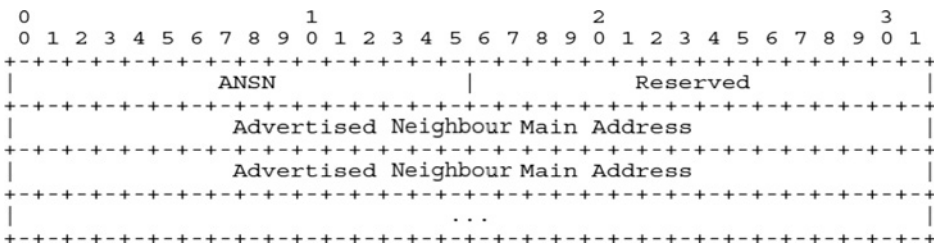



**Fig. 3** *TC message format*

• *Incorrect traffic generation (ITG)*: This attack generates incorrect traffic although the messages are correct. There are two types: (i) 'Replay' attack, in which an intruder node forwards obsolete messages generated in the network and (ii) 'Relay' attack (Wormhole attack) [11], difficult to detect, in which two or more nodes that collaborate to create a tunnel between them have direct access to the network. Once the tunnel has been created (worm hole), attackers encapsulate their ingoing messages and carry out the exchange of these messages through the tunnel, preventing intermediate nodes from receiving control messages.

### 1.2 COD-optimised link state routing

The proposed extension, called COD-OLSR, aims to detect incorrect message generation, making spoofing more difficult. The COD-OLSR mechanism consists of two phases: coding and verification. These two phases are interrelated, so that the integrity of an encoded message in the sender is checked by the receiver, as shown in Fig. 4. Phase encoding uses binary operations (complement to one and XOR) that transform the information into 32-bit numbers. In this encoding process a series of functions are used allowing three fields to be generated which are then added to the control message header as shown in Fig. 5. These fields are: COD originator address, COD links and COD random, for a total of 12 bytes, so that the overhead introduced is negligible. Each of these fields has a specific functionality. Thus, in order to prevent identity spoofing, the message must generate COD originator address as shown in Fig. 6. This field is created by GenerateCodOA function [see (1)], using the following parameters: the main direction of node (main address), which is the network address that configures to the node, a message sequence number MSN of two bytes and a random number random of 32 bits generated by the function GenerateRandom. This last function generates numbers taking into account the current system and therefore providing greater randomness.

$$\text{GenerateCodOA} = \overline{\text{OA} \oplus \overline{\overline{\text{RND}}} \oplus \text{MSN}} \tag{1}$$

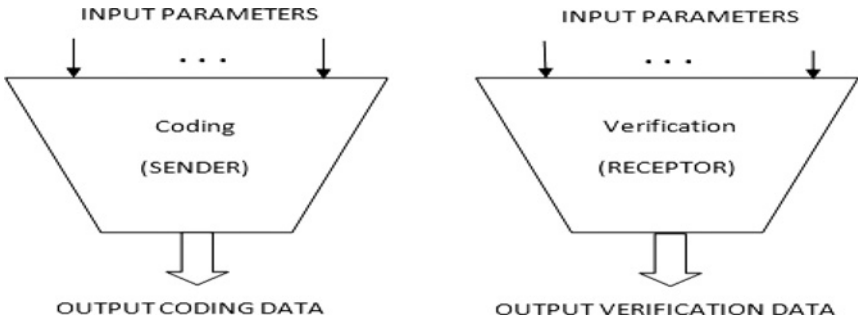



**Fig. 4** *Coding and verification process*

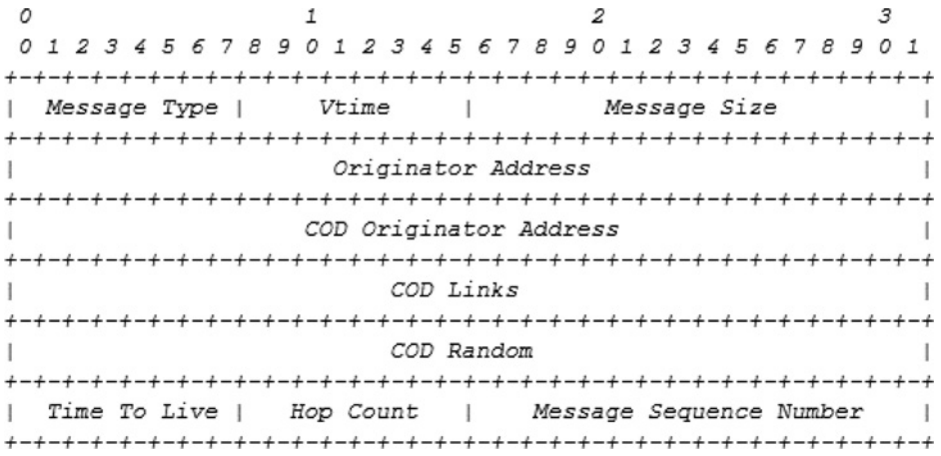



**Fig. 5** *COD-OLSR message header format*

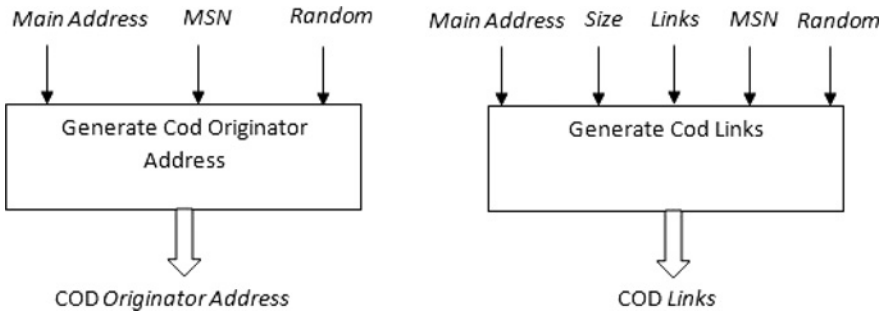



**Fig. 6** *COD originator address and COD links generation*

where OA, originator address; RND, random number generated by GenerateRandom function; MSN, message sequence number.

Similarly, in order to prevent link spoofing, a COD links field is used, which is encoded using the GenerateCodLink function as shown in Fig. 6. This field is constructed by combining several parameters [see (2)]: the main direction of node (main address), the value of the message size (size, 16 bits), the same random number random mentioned above, and a list of 32-bit values associated with the content of control messages.

$$\text{GenerateCodLink} = \overline{\text{OA} \oplus \overline{\overline{\text{RND}}} \oplus \text{MSN} \oplus \text{SIZE} \oplus \overline{\overline{\text{LINK}}}} \tag{2}$$

where SIZE, packet size (bytes); LINK, 32-bit value encoded with the elements of the list.

This list of 32-bit values associated with the content of control messages can be associated with the addresses of the neighbours if it is a HELLO control message or addresses of MS in the case of TC control messages.

This list is encoded from $N$ to 1 [see (3)], that is, several items are transformed into one through the binary operations mentioned above.

$$\text{LINK} = \begin{cases} \text{link}_{i-1} \oplus \overline{\text{link}_i} & \text{if } i \text{ is even} \\ \text{link}_{i-1} \oplus \text{link}_i & \text{if } i \text{ is odd} \end{cases} \tag{3}$$

where $\text{link}_{i-1}$, the previous value of the list; $\text{link}_i$, the current value of the list.

Finally, in order to increase protection, the COD random field is provided. This field is created with the GenerateCodRandom function [see (4)]. As shown, the parameters used are: originator address, the previous message sequence number MSN, random number random decoded and a list of values (IP address of nodes) that are the key of COD-OLSR. This list of 32-bit values represents the current topology with regard to the node that generates the control message, which also has an encoding $N$ to 1.

$$\text{GenerateCodRandom} = \overline{\text{OA} \oplus \text{RND} \oplus \overline{\overline{\text{MSN}}} \oplus \text{LINK}} \tag{4}$$

As mentioned above, the attack detection is performed in the receiver (in the verification phase), which checks that the fields encoded in the sender have not been altered, ensuring the integrity of messages.

In the example shown in Fig. 7, we see how both the sender and the receiver nodes take the topology into account. The sender node $C$ sends a HELLO control message to node $D$. One hop neighbours of $C$ are topology = $\{A, B, D\}$. When

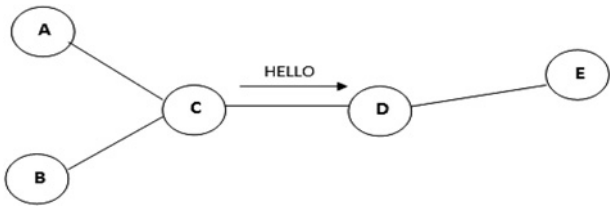



**Fig. 7** *Scheme of topology coding*

the receiver $D$ receives this message HELLO the topology of sending node $C$ must be calculated in the verification process. Thus, it calculates the two-hop neighbours, with the intermediate node being $C$. To avoid problems of inconsistency (since the sending node topology may change after sending the message), we used a technique of synchronisation so that the elapsed period is very small (in the order of milliseconds) from the message coding phase in the sender to the verification in the receiver. In the verification process, the function RecoverCodRandom is used [see (5)].

$$\text{RecoverCodRandom} = \text{OA} \oplus \overline{\text{MSN}} \oplus \text{LINK} \oplus \overline{\text{COD}} \tag{5}$$

Thus, the receiver calculates which neighbours the sender has. Once the receiver has found the topology of the sending node, the reverse process is performed: first the random number random using this function is calculated, which retrieves the original random number generated by the sender, by combining the following parameters: COD random, originator address, MSN and topology. Then with this random number random and with information of message received by the receiver, it calculates COD originator address and COD links as seen in the coding phase. If the calculated values do not correspond to the fields COD originator address and COD links contained in the message, the receiver decides that an attack has taken place and it discards the message.

Fig. 8 shows a diagram for the generation of COD random. The most important feature of this diagram is that it uses the encoding of the current topology of the sending node, so that the suspected attacker, ignoring the dynamic topology of the network, although he can see the message, cannot interpret it. If the topology of the sending node does not change, the attacker can exploit this situation by monitoring the network and then performing the spoofing. As a security measure, for the coding of the three fields COD originator address, COD links and COD random the message sequence number MSN is used.




doi: 10.1049/iet-ifs.2010.0085

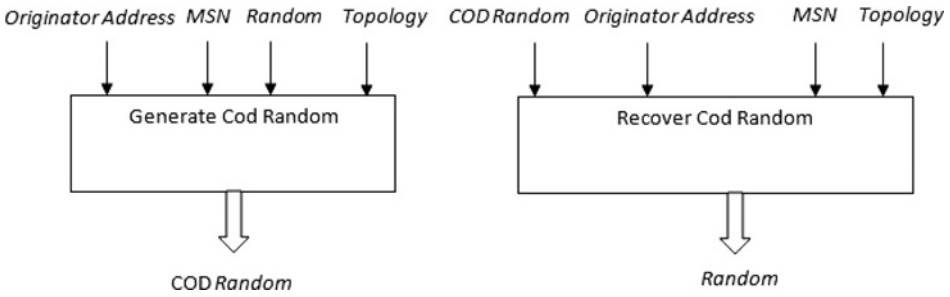



**Fig. 8** *Generate and recover of random*

### 1.3 Simulation

In this section we analyse the behaviour of COD-OLSR in face of OLSR. We also check the efficiency of COD-OLSR with different percentages of attackers and in different situations to see how the most representative performance metrics, such as overhead, system failure tolerance, delivered data packet ratio and throughput evolve.

### 1.4 Performance metrics

In this section we define the most important performance metrics used in the simulation:

- *Overhead in number of packets*: Relationship between the total numbers of transmitted control packets by the nodes of the network and the number of delivered data packets to their destinations.
- *Overhead in number of bytes*: Relationship between the total numbers of transmitted control bytes and delivered data bytes.
- *System failure tolerance*: This metric takes into account the number of control messages that are discarded by COD-OLSR.
- *Delivered data packet ratio*: Relationship between number of sent packets and the number of delivered packets successfully.
- *Throughput*: Volume of work or information flowing through a system. Is calculated by dividing the total number of bits delivered to the destination by the packet delivery time.

## 2 Results

In the simulation we utilised the Network Simulator NS-3 [12]. We have carried out two kinds of experiments: the first experiment compares the overhead between COD-OLSR and OLSR. The second one consists of a study of COD-OLSR against various attacks. In this second experiment, the results have been obtained according to various performance metrics. In both experiments the following assumptions were considered: (i) the intruder nodes do not carry out simultaneous attacks in HELLO and TC control messages, (ii) the attack in question is either identity spoofing or link spoofing, not both kinds of attacks in the same message. The following features are shared by both experiments: we have used an area of $1000 \times 1000$ m$^2$ and a simulation time of 30 s. Four data sessions have been considered (always the same source–destination pairs). The application used for data generation has been constant bit rate, in which four packets of 64 bytes of data per second are sent, that is a data transmission rate of 2048 bps. The used mobility model has been random waypoint with a speed between a minimum of 0 and a maximum of 5 m/s, with pause time of 5 s. Network devices were configured according to the standard IEEE 802.11b with a transmission range of 150 m.

### 2.1 Comparison of the overhead between COD-OLSR y OLSR

In this comparison a variable density of nodes between 20 and 100 nodes has been used and we compared the overhead of the two protocols without considering any type of attack in order to see if the COD-OLSR overhead is insignificant. We draw the comparison with both the number of packets and of bytes because they do not behave in the same way, since the first considers the number of messages per packet, while the second refers to the total size of the packet in bytes.

### 2.2 Overhead in number of packets

Fig. 9 shows the comparison of the overhead in the number of packets between COD-OLSR and OLSR. We see that in both protocols the overhead is very similar; nevertheless, in very dense networks COD-OLSR has a slight overhead compared to OLSR.

### 2.3 Overhead in number of bytes

In Fig. 10, we see how the overhead in the number of bytes is a bit higher in COD-OLSR than in OLSR when we increase

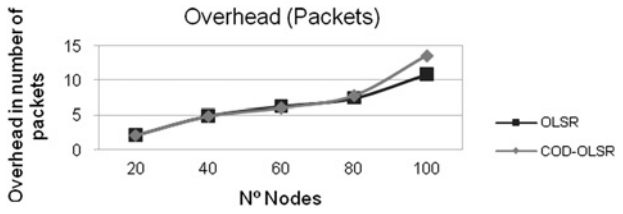



**Fig. 9** *OLSR against COD-OLSR according to overhead in packets*

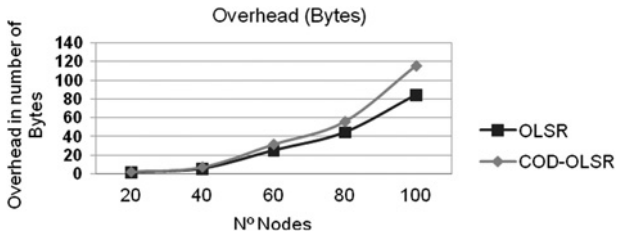



**Fig. 10** *OLSR against COD-OLSR according to overhead in bytes*

the density of nodes. This is because the header control messages of COD-OLSR have three more fields (4 bytes each one), so that the protocol has to process 12 bytes more.

### 2.4 Study of COD-OLSR

In this section we analyse the behaviour of the proposed extension of OLSR with attacks of different characteristics, to verify that the stability of the protocol is not altered.

### 2.5 System failure tolerance

The system failure tolerance is the fraction of lost control messages (not processed by the algorithm). In this experiment 100 nodes for the representation of the graph are used.

In Fig. 11, we see that the system failure tolerance depends more on the number of intruders than the number of attacked messages. We appreciate in this graph that if there is a 100% message attack with 20% intruders, packet loss is slightly over 12%. We also see that, when the number of intruders decreases to 5%, the loss of messages does not exceed 2%, whatever the percentage of attacked messages may be. These results make us see that COD-OLSR has a good system failure tolerance, providing stability to OLSR.

### 2.6 Delivered data packet ratio

In Fig. 12, we see the delivered data packet ratio, taking into account that malicious nodes attacks to control messages are 100%. Thus, we observe the behaviour of the system in the most extreme case. We note that the ratio depends on increasing the density of nodes. In sparse networks (20 nodes) the resulting line is nearly uniform, held within the 20% ratio, regardless of the number of attackers, while in very dense networks (100 nodes), the ratio decreases from 70 to 20% when we increase the number of intruders, since the attacks provoke incorrect routing.

### 2.7 Throughput

The throughput behaves similar to the ratio, as shown in Fig. 13. These results make us see that both throughput and

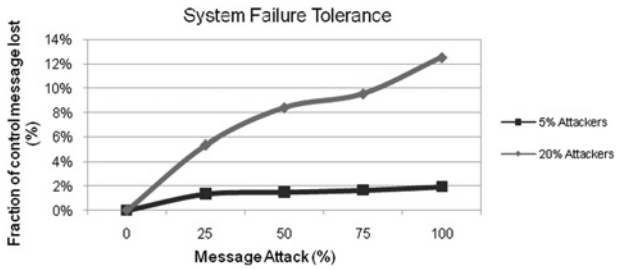



**Fig. 11** *System failure tolerances (COD-OLSR)*

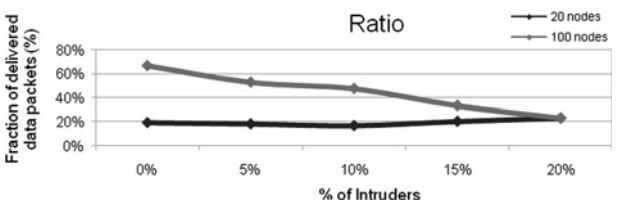



**Fig. 12** *Delivered data packet ratio (COD-OLSR)*

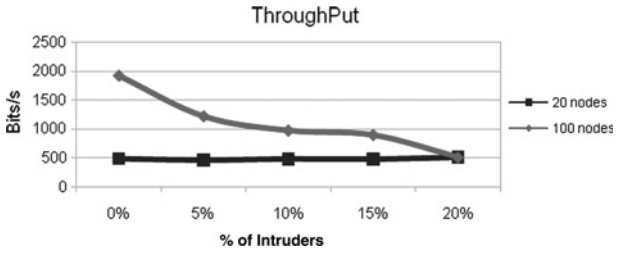



**Fig. 13** *Throughput (COD-OLSR)*

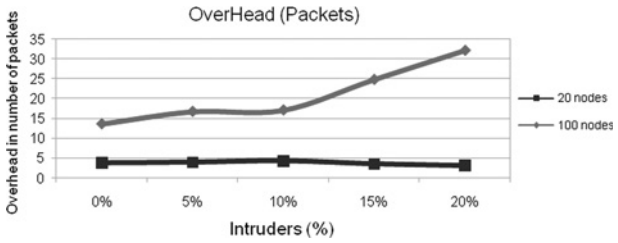



**Fig. 14** *Overhead in packets (COD-OLSR)*

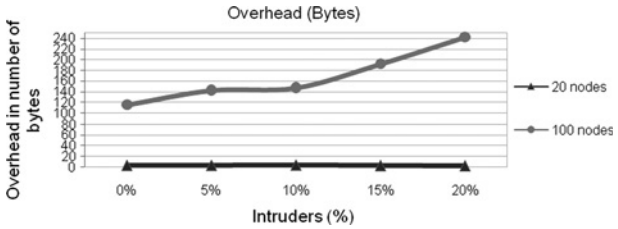



**Fig. 15** *Overhead in bytes (COD-OLSR)*

delivered data packet ratio metrics are closely related, but they use different scales.

### 2.8 Overhead (packets)

In Fig. 14, we analyse the behaviour of overhead in number of packets according to the density of network nodes with 100% of attack messages. In sparse networks (20 nodes) it does not overcome five packages, and so it presents a uniform line whatever the percentage of attacks is. When, we increase the number of intruders in very dense networks (100 nodes), the overhead in the number of packets increases dramatically. We see that with 20% of attackers, overhead reaches a value of more than 30 packets.

### 2.9 Overhead (bytes)

Under the same conditions as in the previous case, that is, Fig. 14, the behaviour of overhead in the number of bytes is very similar, as shown in Fig. 15. In sparse networks, the overhead in number of bytes is practically negligible whatever the percentage of attackers. In contrast, it increases considerably in very dense networks when the number of attackers increases.

## 3 Conclusions

Routing protocols for MANETs do not take into account the hostile environments, so that security extensions should be added. We analyse OLSR, one of the most important routing protocols, which does not have security in its specification, presenting an extension to this protocol, called COD-OLSR,




doi: 10.1049/iet-ifs.2010.0085

which protects it from incorrect message generation, both identity spoofing and link spoofing. One of the most important characteristics of this extension is that it takes into account the current topology of the sending node, so that the receiver of the message can verify the integrity of control messages. The obtained results show that COD-OLSR has a slight overhead, but without being detrimental for the performance of OLSR protocol, ensuring integrity as seen in the performance metrics. We also see that this method is an alternative to security using cryptography, in which there is a high overhead. For further research, we are working on adapting our framework to other kinds of vulnerabilities, such as 'Replay' attacks and Wormhole attacks.

## 4 Acknowledgments

This work was supported by the Ministerio de Ciencia e Innovación (MICINN) through the Projects TEC2007-67129/TCM and TEC2010-18894/TCM and the Ministerio de Industria, Turismo y Comercio (MITyC) through the Project Avanza Competitividad I + D + I TSI-020100-2010-482.

## 5 References

1 IETF: 'Mobile ad-hoc networks (MANET) working group'. Available at http://www.ietf.org/html.charters/manet-charter.html

2 Clausen, T., Jacquet, P.: 'Optimized link state routing protocol (OLSR)'. IETF RFC3626, 2003, available at http//www.ietf.org/rfc/rfc3626.txt

3 Halfslund, A., Tonnesen, A., Rotvik, R.B., Andersson, J., Kure, O.: 'Secure extension to the OLSR protocol'. OLSR Interop and Workshop, 2004

4 Adjih, C., Clausen, T., Jacquet, P., Laouiti, A., Mühlethaler, P., Raffo, D.: 'Securing the OLSR protocol'. Proc. Med-Hoc-Net, Mahdia, Tunisia, June 2003

5 Raffo, D., Clausen, T., Adjih, C., Mühlethaler, P.: 'An advanced signature system for OLSR'. ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN-04), Washington, DC, USA, October 2004

6 Raffo, D., Adjih, C., Clausen, T., Muhlethaler, P.: 'OLSR with GPS information'. Proc. 2004 Internet Conf. (IC 2004), Tsukuba, Japan, October 2004

7 Papadimitratos, P., Haas, Z.J.: 'Secure link state routing for mobile ad hoc networks'. Proc. 2003 Symp. Applications and the Internet Workshops (SAINT'03 Workshops), DC, USA, January 2003, pp. 379–383

8 Nait-Abdesselam, F.: 'Detecting and avoiding wormhole attacks in wireless ad hoc networks', *IEEE Commun. Mag.*, 2008, **46**, (4), pp. 127–133

9 Abusalah, L., Khokhar, A., Guizani, M.: 'A survey of secure mobile Ad Hoc routing protocols', *IEEE Commun. Surv. Tutor.*, 2008, **10**, (4), pp. 78–93

10 Kannhavong, B., Nakayama, H., Kato, N., Jamalipour, A., Nemoto, Y.: 'A study of a routing attack in OLSR-based mobile ad hoc networks', *Int. J. Commun. Syst. (IJCOMSYS)*, 2007, **20**, (11), pp. 1245–1261

11 Hu, Y.C., Perrig, A., Johnson, D.B.: 'Wormhole attacks in wireless networks', *IEEE J. Sel. Areas Commun. (JSAC)*, 2006, **24**, (2), pp. 370–380

12 The ns-3 network simulator. Available at http://www.nsnam.org

*Journal of Ubiquitous Systems & Pervasive Networks*
*Volume 3, No. 1 (2011) pp. 35-40*

# An Extension Proposal of D2HCP for Network Merging

**Luis Javier García Villalba[a*], Julián García Matesanz[b], Ana Lucila Sandoval Orozco[a], José René Fuentes Cortez[a]**

[a] *Grupo de Análisis, Seguridad y Sistemas (GASS)*
*Departamento de Ingeniería del Software e Inteligencia Artificial (DISIA)*
*Facultad de Informática, Despacho 431*
*Universidad Complutense de Madrid (UCM)*
*Calle Profesor José García Santesmases s/n*
*Ciudad Universitaria, 28040 Madrid, Spain*
*E-mail: {javiergv, asandoval, jrfuente}@ fdi.ucm.es*

[b] *Grupo de Análisis, Seguridad y Sistemas (GASS)*
*Sección Departamental de Sistemas Informáticos y Computación*
*- Lenguajes y Sistemas Informáticos y Ciencias de la Computación e Inteligencia Artificial -*
*Facultad de Ciencias Matemáticas, Despacho 310-F*
*Universidad Complutense de Madrid (UCM)*
*Plaza de Ciencias, 3*
*Ciudad Universitaria, 28040 Madrid, Spain*
*E-mail: julian@sip.ucm.es*

## Abstract

Mobile Ad Hoc Networks (MANETs) consist of mobile nodes equipped with wireless devices. They do not need any kind of pre-existent infrastructure and are about self-managed networks. MANETs enable communication between mobile nodes without direct links and across multihop paths. In order for correct operation of the routing protocols, Mobile Ad Hoc Networks, have to assign unique IP addresses to the MANET devices. Furthermore, the address assignment is an important issue when dealing with MANET networks because the traditional approaches are not applicable without some changes, has to provide new protocols for the address auto-configuration. These schemes must take into account the properties that MANETs such as dynamic topology, limited resources or lack of infrastructure. In this paper, we propose an extension of D2HCP, a stateful scheme for dynamic allocation of IP addresses in MANETs. This extension proposal will allow the network merging not covered by its predecessor.

***Keywords:*** *Mobile Ad Hoc Networks, Auto-Configuration, Network Merging, D2HCP, Extension.*

## 1. Introduction

A Mobile Ad Hoc Networks (MANET) consists of a set of mobile nodes equipped with wireless devices. The nodes establish direct links with every node that is within its transmission range and thereby enable communication between nodes without direct links and across multihop paths. This kind of computer network is characteristic for its dynamic network topology, lack any fixed infrastructure or administration, limited bandwidth and energy saving capacity. They are an important feature within the field of computer networks.

One of the most important issues related to these type of networks is the routing [1]. In order to provide routing in MANETs, it is necessary that the nodes can be uniquely identified across an IP address with the only premise that this address must be different than the one of any other node in the network.

The process of configuration [2] is the set of steps through which a node obtains its IP address within the network.

[*] Corresponding author. Tel.: +34 91 394 76 38
Fax: +34 91 394 75 47; E-mail: javiergv@fdi.ucm.es


DOI: 10.5383/JUSPN.03.01.007

There are two mechanisms to set addresses: stateless and stateful.

The stateless address configuration proposes its own node to be the one in changed of generating its IP address. The address is obtained from the concatenation of a network well-known prefix and the theoretically unique number inside the network generated by the node.
This mechanism may require the inclusion of a module in responsible for verifying the uniqueness of the generated address called Duplicate Address Detection (DAD) [3] [4] 5]. On the other hand, the Stateful address configuration is based on using servers which control and assign addresses to all the nodes of the network.
Dynamic Host Configuration Protocol (DHCP) [6] is an example of Stateful configuration. However, because of the multi-hop nature of the mobile ad hoc networks, this protocol cannot be applied directly.
There are numerous works that present proposals for address configuration in a Mobile Ad Hoc Network using the Stateful [7] [8] [9] and Stateless [3] [10] [11] mechanism.
A comprehensive review of the main stateful (and stateless) auto-configuration protocols can be found in [12].
D2HCP [13] is a new proposed stateful autoconfiguration scheme that uses the OLSR proactive routing protocol for synchronization and guarantees unique IP address under a variety of network conditions including message losses and network partitioning. This paper presents an extension of D2HCP that includes the network merging not covered by its predecessor.
This work is structured in 4 sections; the first one is the present introduction. Section 2 contains the specification of D2HCP, a proposal about IP address auto-configuration for Mobile Ad Hoc Networks. Section 3 presents the proposal of extension of D2HCP to allow network merging. Finally, Section 4 shows the main advantages of the developed new protocol as well as potential future extensions to the study.

## 2. D2HCP

The Distributed Dynamic Host Configuration Protocol (D2HCP) is an auto-configuration protocol that manages the joining and departing of nodes in MANET. Nodes have a synchronization system is based on the OLSR [14] routing protocol. Thanks to this mechanism, the synchronization is done passively, monitoring the mentioned routing protocol, thus no overhead is generated in the network traffic compared to that one generated by the OLSR protocol. Due to the fact that all the nodes are responsible for managing the joining of any new node to the network, this process can be done quickly.
A node that wishes to join a network tries to contact any node still belonging to it, and may receive several responses from multiple nodes. This makes the chances of successfully joining the network high, because of the high availability and redundancy that the distributed management disposes.

### 2.1. Message Format

All the sent messages are packed in the protocol with the format shown in the Figure. 2.

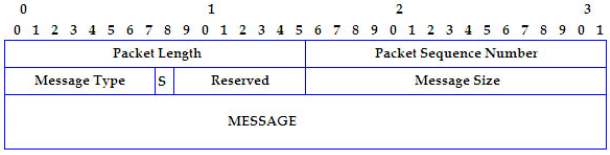



**Figure 2. Packet from D2HCP protocol.**

These messages will be encapsulated in turn with the headers corresponding to the MAC or TCP/IP, depending on the type of message to be included in the MESSAGE field.

#### *2.1.1. Packet Header*

The first row of the Fig. 2 contains the fields of the packet header.

- Packet Length: Packet length, including the header (2 bytes).
- Packet Sequence Number: Sequence Number (2 bytes). In each different message which is sent by node, this field is increased by one. It helps in being able to detect duplicated packets.

#### *2.1.2. Message Header*

The second row of the Fig. 2 is the header of each one of the protocol message:

- Message Type: It has 1 byte of size.
- S (Security): Reserved for security implementation (1 bit).
- Reserved: Reserved for functionality future extensions (7 bits).
- Message Size: It consists of 2 bytes.

Next the format of each kind of message is shown, include in the MESSAGE field.

#### *2.1.3. SERVER_DISCOVERY*

The Figure 3 shows the SERVER_DISCOVERY message format.

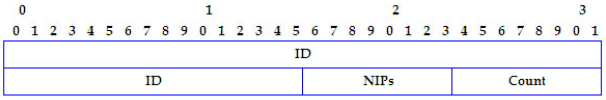



**Figure 3. SERVER_DISCOVERY message format.**

- ID: Node Identification (6 bytes).The node has to choose the MAC address from one of its interfaces, if it has more than one. This identity field has the same value for every SERVER_DISCOVERY and SERVER_POLL message emitted by the node although it was doing since different interfaces.
- NIPs: Amount of IP addresses solicited by the node. It will be equal to interface number of client node (1 byte).
- Count: Number of times that the SERVER_DISCOVERY petition has been tried (1 byte).

#### *2.1.4. SERVER_OFFER*

Figure 4 shows the SERVER_OFFER message format.

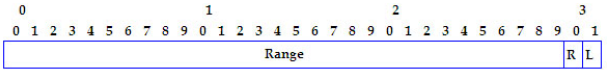



**Figure 4. SERVER_OFFER message format.**

- Range: Number of IP addresses offered (30 bits).
- Ready (R): It indicates whether the node that offers the IP addresses is ready to assign them or only communicates their existence but at this point can not assign them (1 bit).
- Local (L): It indicates whether the range offered is sending node, or else be asked to turn to a third node (1 bit).

#### *2.1.5. SERVER_POLL*

The Figure 5 shows the SERVER_POLL message format.

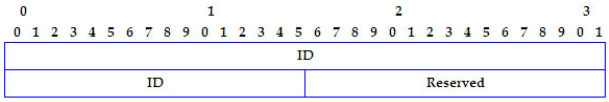



**Figure 5. SERVER_POLL message format.**

- ID: Node identification (6 bytes). It is the same identification as the elected in the SERVER_DISCOVERY message.
- Reserved: Reserved for future implementations (2 bytes).

#### *2.1.6. IP_ASSIGNED*

The Figure 6 shows the IP_ASSIGNED message format.

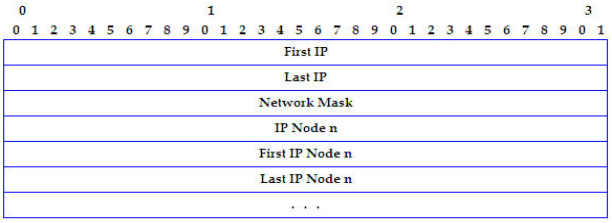



**Figure 6. IP_ASSIGNED message format.**

- First IP: IP address initial of free address block (4 bytes).
- Last IP: IP address end of free address block (4 bytes).
- Network Mask: It consists of 4 bytes.
- IP Node n, First IP Node n, Last IP Node n: It represents a entry of free block table for all the nodes in the network. Each node is represented by these fields of 4 bytes, each one being: the node IP address, the initial IP address and the final from its free address block, respectively.

#### *2.1.7. IP_RANGE_REQUEST*

The Figure 7 shows the IP_RANGE_REQUEST message format.

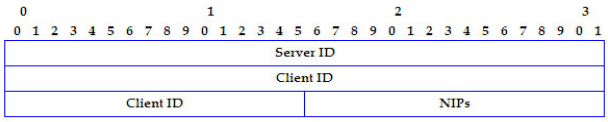



**Figure 7. IP_RANGE_REQUEST message format.**

- Server IP: Server address that requests the range for client (4 bytes). It can be different from IP address of sender message, by treating multi-hop networks.
- Client ID: Client node identification. It has the same value as the ID field of the SERVER_DISCOVERY and SERVER_POLL message (6 bytes).
- NIPs: Solicited IP Address Number (2 bytes).

#### *2.1.8. IP_RANGE_RETURN*

Figure 8 shows the IP_RANGE_RETURN message format.

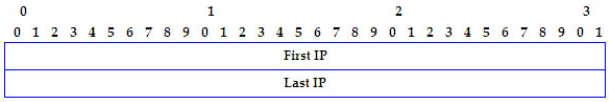



**Figure 8. IP_RANGE_RETURN message format.**

- First IP: The initial IP address of free address block (4 bytes).
- Last IP: The final IP address of free address block (4 bytes).

### 2.2. Timers

The wireless and mobile nature of MANET means that there are situations in the networks where messages are lost, or delayed in getting to its destination more than the estimated time.

We expose below a series of timers used to solve such situations:

- SERVER_DISCOVERY_TIMER: After sending the message, the client node will start this timer when it gets to the state WAITING_REPLY. During this time the node is waiting SERVER_OFFER message of possible nodes close to belonging to a network.
  The longer the timer runs, the more time it will dedicate to receiving messages of this kind, thus there will be more to process and, therefore, it is easier to get an address block. But it also means increasing the latency to obtain an IP address.
  If this timer expires and the client node has not received any SERVER_OFFER, there has been a message loss, or perhaps there are no server nodes, it will send a new SERVER_DISCOVERY. This action is repeated SDISCOVERY_MAX_RETRY times and, if it goes on without receiving messages, it will initiate its own network.
- SERVER_OFFER_TIMER: After the message SERVER_OFFER, the node goes to the state WAITING_POLL, and initiates this timer. When it expires, the state will change to IDLE.
- SERVER_POLL_TIMER: As soon as the message is sent SERVER_POLL, the node client will wait for the message IP_ASSIGNED by the time that this timer should determine. If this message does not come, the SERVER_POLL will be re-transmitted up to a maximum number of attempts, defined as SPOLL_MAX_RETRY. If the maximum number of attempts is exceeded, it will begin the configuration process again.
- IP_RANGE_REQUEST_TIMER: The server node who sends a message IP_RANGE_REQUEST to another node of the network initiates this timer at that moment. As with the SERVER_POLL_TIMER if this timer expires, the message IP_RANGE_REQUEST will be forwarded up to RREQUEST_MAX_RETRY attempts.
- ACCEPTED_OFFER_TIMER: This timer is activated after sending a message IP_ASSIGNED or a message IP_RANGE_RETURN. During this time, the server node cannot reply to requests of type SERVER_POLL or IP_RANGE_REQUEST. This restriction will get up after the timer expires (the offer expired without being accepted), or on having detected that a node with the first IP address of the offered ones has entered the network (the offered address block was accepted).
  It is necessary to bear in mind that although it could not assign IP address, the server node will keep on answering requests SERVER_DISCOVERY giving the value 0 to the field R (READY) in the message SERVER_OFFER. In this way, the node client is informed of the existence of the server, although it should not be capable of assigning IP address immediately.

- NODE_DOWN_TIMER: When OLSR erases the route towards a node, it is not eliminated immediately from the Free_IP_Blocks table. In its place, this timer is initiated. If before the timer expires it turns to discover a route towards the node, that means that that it disappeared momentarily, but it did not leave the network. Therefore, the elimination is cancelled from the Free_IP_Blocks table. In case the timer expires and a route has not been recovered, it is given to the node to be lost and its entry is eliminated from the Free_IP_Blocks table, updating those who correspond.
- INIT_TABLE_TIMER: On having received the Free_IP_Blocks auto-configuration table in the message IP_ASSIGNED, the client node activates this timer. During that time, the table contains nodes so that OLSR does not have a well-known route yet. After the timer expires, the nodes from the Free_IP_Blocks table do not have entry in the OLSR route table are verified: these nodes are eliminated (updating the entries that correspond), since they are nodes that belonged to the network on having received the table, and have left it before OLSR knew of its existence.
- INIT_ASSIGN_TIMER: This timer is used as much by the node client as by the server when they have received or assigned, respectively, an IP address block. That is to say, the server initiates it, on having verified the node entry with the first IP address of the block offered in the IP_ASSIGNED message, and the client initiates it after receiving the IP_ASSIGNED message and to configure its address.

Thus there has been time so that the whole network could update its Free_IP_Blocks table before more changes take place. During that time, they will ignore SERVER_POLL or IP_RANGE_REQUEST messages, although they will reply to the SERVER_DISCOVERY.

- NODE_DOWN_ASSIGN_TIMER: When an already configured node detects the departure of another one, and it verifies that it is its turn to gather the IP address that remains free, it starts its timer. More concretely, the timer will be activated when the elimination of the OLSR routing table is detected, that is to say, it will be activated at the same time as the NODE_DOWN_TIMER timer.
  Until it does not expire, the node will ignore the SERVER_POLL and IP_RANGE_REQ requests. This way, a margin of time will happen to insure itself that all the nodes in the network detect the mentioned departure and update its Free_IP_Blocks table, before assigning them to some another new node. Therefore, the duration of this timer must be greater than that of the NODE_DOWN_TIMER to make sure that the rest of nodes in the network not only have detected the elimination of a route but they have eliminated it from the Free_IP_Blocks table.
  The node will keep on replying to the SERVER_DISCOVERY messages fixing the value 0 in the R field from the SERVER_OFFER message.
- SLEEP_TIMER: Used timer by a client node when it detects nearby nodes belonging to some network, but that are not in disposition to assign IP addresses in this moment. This way, it gives a margin of time to allow the processes to end preventing them from assigning address.

### 2.3. State of Nodes

Depending on if they are in the process of joining the network, or if they already belong to this one, two types of nodes are differed: client and server. In the following paragraphs are showed and explained the states that govern the behavior of both types of nodes. The state, in which the node is found, changes on having produced sending or receptions of messages, or when determined timers expire.

#### *2.3.1. Server Node*

We call all the nodes in the network that are configured correctly; server node, that is to say, they possess a valid IP address with which they can communicate with the rest of nodes, and a free IP address block. With this free address block they facilitate the access to the new nodes, which we will call clients. Two types of states exist: ready or not_ready.
From any state, the server node is in expectation of SERVER_DISCOVERY messages. It will always reply with a SERVER_OFFER message, but depending on whether the current state of the node is of type ready or not_ready, it will answer giving to the field R (READY) the value 1 ó 0. This way, a node always announces its presence, although at this precise moment it should not be capable of assigning IP addresses to a client.

- Any state (ready): From any state of type ready, the server node will respond to a SERVER_DISCOVERY message with one of type SERVER_OFFER (R field with value 1). That will ensure that the server goes to the state Wait SERVER_POLL. A node can reply at the same time SERVER_DISCOVERY to requests from different nodes, and to be in expectation of any of the correspondents SERVER_POLL. Also they will be answered to messages of type IP_RANGE_REQUEST with one of the type IP_RANGE_RETURN. This means that whether the node is in any state ready, it will be give half of its free address blocks to any other node in the network without proper addresses and that it needs to facilitate the join to a client.
- Any state (not_ready): Whilst the node is in a state of type not_ready, it will reply to the SERVER_DISCOVERY messages with a SERVER_OFFER message giving the value 0 to the field R.
- Wait SERVER_POLL: In this state the node waits for a determined time by the SERVER_OFFER_TIMER timer to receive a SERVER_POLL message. This message indicates that the client, the one who sent the SERVER_OFFER has chosen as its server for the process of auto-configuration. After the reception of the message, the node will send a IP_ASSIGNED message to the client in the case of having available addresses locally (the field L of the message SERVER_OFFER had value 1); and it will pass to the Wait IP_Assigned state resolve. If the addresses that it offered were not local, it will have to ask for them from a node in it network with the IP_RANGE_REQUEST message; and it will pass into the state Wait IP_RReturn. In case any SERVER_OFFER message are not received before the timer expires, the node will pass to be in the state IDLE.
- Wait IP_ASSIGNED resolved: In this state, of type not_ready, the node is waiting to find out if the client node correctly received the address block offered by an IP_ASSIGNED message, or through an IP_RANGE_RETURN message and an intermediary. The result can be that the client has been configured correctly, or that it has not received the address block. The above-mentioned can take place for several motives, since they can be problems of interferences in the message reception, moving of the client node out of the coverage range, so on. If a new node appears in the network using the first IP address from the block offered in the message IP_ASSIGNED or IP_RANGE_RETURN, it means that

the client node stopped being configured. In this moment the server node changes its state to Init steal window. If the node was not capable of finishing the process of auto-configuration, the ACCEPTED_OFFER_TIMER timer will expire. In this case the node passes to the state IDLE.

- Init time window: This state serves to give a time margin that allows all the nodes in the network to be capable of detecting the join of the client recently configured, before dividing again the own IP free address block. If this margin was not happening, and new requests would be attended immediately, synchronization problems might happen if other nodes were detecting the new incorporations to the network in an incorrect order.
- Wait ip_return: In this state the node is hanging out for the reception an IP_RANGE_RETURN message. When it receives this message, in which a node in the network indicates it a block that can offer the client in waiting, it will send an IP_ASSIGNED message to the client. After this, the node will have ended its function as server, and will pass to the state IDLE. If the IP_RANGE_RETURN message is not received before the timer IP_RANGE_REQUEST_TIMER expires, it will turn to try the request sending again an IP_RANGE_REQUEST message a maximum number of times RREQUEST_MAX_RETRY. These successive attempts are sent in every occasion to a different node. If the limit of attempts is exceeded, then the node will desist and change its state to IDLE.
- IDLE: This is the state of rest, or in that the node is idle. When it is in this state, the node does not undergo any operation related to the auto-configuration. It is therefore treated as a state of waiting.
- Node down time window: This node provides a time margin when the node must gather the IP address from a node that has left the network. More precisely, the node changes to this state on having detected that it has lost the route towards a node of whose address block is responsible. This transition is done from any other state, be already of type ready or not_ready. After the time determined by NODE_DOWN_ASSIGN_TIMER, the node will return to the state of rest IDLE. It is not to be confused this timer with the NODE_DOWN_TIMER. Although they begin at the same time on having detected the same event, the involved processes are independent.

#### *2.3.2. Client Node*

The client node is provided with states of type not_ready, explained in the states of the server node. As if it was treated of an already configured node, the node that is in process of configuration of its IP address reply to SERVER_DISCOVERY requests with SERVER_OFFER messages. In these messages the value 0 is given to the R field, since the node is not in disposition to assign IP addresses, only he tries to announce its presence.

- Initial state: in which the begins process of auto-configuration. If the number of attempts is minor to the maximum, SDISCOVERY_MAX_RETRY, then the node emits a SERVER_DISCOVERY message for each of its network interfaces which are going to use the network MANET. It changes its state to Receive SERVER_OFFER. If it has gone over at the limit of attempts, then the node desists from his intention from finding a network to which to join and creates a new one. It will pass to be a node server, beginning in the state IDLE.
- Receive SERVER_OFFER: It is treated as a state of waiting, during which the SERVER_OFFER messages of next possible nodes are gathered. On having ended the waiting, determined by SERVER_DISCOVERY_TIMER, the responses are processed. If there has not been received any response SERVER_OFFER, it is turned to the initial state. If there is some offer, the number of them with the value 1 in the field R is verified. If between the offers none had the bit R to 1, it means that that there are nearby nodes belonging to a network, but at the moment they are not capable of assigning IP addresses. Therefore, the node passes to the state Sleeping.
  If there was some offer with the bit R to 1, the server are sorted by preference and there is sent a SERVER_POLL message to the first one of them. Into this case the node changes his state to Wait IP_ASSIGNED.
  This state is not of any of the types explained in the state of a client node. This means that it does not reply to SERVER_DISCOVERY messages, since at the moment it does not know if there is any network nearby which to join.
- Sleeping: The node interrupts his attempts of joining the network during the time determined by the SLEEP_TIMER timer. This is like that because there have been received SERVER_OFFER messages of nearby nodes that at the moment are not capable of assigning IP addresses, and what is claimed that after this time they are already capable of facilitating the join to the network. After the timer expires, the node will return to the initial state
- Wait IP_ASSIGNED: In this state, the client is in expectation of an IP_ASSIGNED message on the part of the server to whom the message SERVER_POLL was sent. If the awaited message does not come, the SERVER_POLL_TIMER timer expires. In this case, it will turn to try to send a SERVER_POLL to the following server of the list generated after the end of the SERVER_DISCOVERY_TIMER timer. If the list of servers is ended, or it goes over the limit of attempts SPOLL_MAX_RETRY, the node returns to the initial state. On having received the IP_ASSIGNED message, the node configures its address (or addresses, in case of having several network interfaces). At this moment, it already takes part normally in the network, and passes to be a node server.

The state with the one begins its behavior as server is the Init time window.

## 3. Extension Proposal of D2HCP

Our extension proposal requires that the following changes are made to the protocol D2HCP:

1. Two new messages added:
   - ClientSynchronization: When two adjacent nodes begin the auto-configuration process and another node that has progressed on this task, doesn't exists, they send a synchronization message to establish which of them will have priority to initiate the network.
   - Hello: Message sent periodically to detect possible network merging.
2. States have been modified:
   - Searching Server: State indicating the node has sent the Server_Discovery message.
   - Waiting Allocation: It enters this state when the client node has found a server that can provide a valid address.
   - Suspended: A client node enters this state when none of the nodes that have responded to the Server_Discovery message have completed the configuration process

. Synchronizing: When two adjacent nodes have not found any node that has started or ended the configuration setup, they go to start a new network. In this state the decision is made as to which nodes have priority in this process.

## 4. Conclusion

The D2HCP protocol is a stateful approach and so it does not require Duplicate Address Detection (DAD), which reduces the overheads. Additionally, monitoring the routing protocol simplifies the synchronization process. In this paper an extension of D2HCP protocol is defined. Among the improvements which it contemplates, exist the following:

- Network Merging.
- Support for the failure of client-server links during the configuration process.
- Improvements in the network Initiation process achieving node synchronization which they wish to initiate the network concurrently.
- Improvement in latency.

Topics for future research include the development of QoS and security extensions.

## Acknowledgments

This work was supported by the Ministerio de Industria, Turismo y Comercio (MITyC, Spain) through the Project Avanza Competitividad I+D+I TSI-020100-2010-482, the Ministerio de Ciencia e Innovación (MICINN, Spain) through the Project TEC2010-18894/TCM and the Agencia Española de Cooperación Internacional para el Desarrollo (AECID) del Ministerio de Asuntos Exteriores y Cooperación (MAEC) through the Project MAEC-AECID C/033548/10.

## References

[1] Corson, S. and J. Macker, J.: 'Mobile ad hoc networking (MANET): routing protocol performance issues and evaluation considerations', Internet Engineering Task Force, RFC 2501, 1999.

[2] Bacelli, E.: 'Address Autoconfiguration for MANET: Terminology and Problem Statement', Internet Engineering Task Force, Internet Draft, 2008.

[3] Perkins, C., Malinen, J.T, Wakikawa, R. et al: 'IP address autoconfiguration for ad hoc networks', Internet Engineering Task Force, Internet Draft, 2001.

[4] Weniger, K. PACMAN: Passive Autoconfiguration for Mobile Ad hoc Networks. IEEE J. Sel. Areas Comm. 2005, 507-519. http.dx.doi.org/10.1109/JSAC.2004.842539

[5] Weniger, K. Passive Duplicate Address Detection in Mobile Ad Hoc Networks. In Proceedings of IEEE WCNC 2003, New Orleans, Louisiana, USA, March 2003.

[6] Droms, R.: 'Dynamic host configuration protocol', Internet Engineering Task Force, RFC 2131, 1997.

[7] Nesargi, S. & Prakash, R.: 'MANETconf: Configuration of hosts in a mobile ad hoc network', Proceedings of the IEEE INFOCOM, New York, USA, 2002, pp. 1059-1068.

[8] Mohsin, M. and Prakash, R.: 'IP Address Assignment in a Mobile Ad Hoc Network', in Proceedings of IEEE MILCOM, September 2002. http.dx.doi.org/10.1109/MILCOM.2002.1179586

[9] Thoppian, M. and Prakash, R.: 'A Distributed Protocol for Dynamic Address Assignment in Mobile Ad Hoc Networks', IEEE Transactions on Mobile Computing, 5 (1), January 2006. http.dx.doi.org/10.1109/TMC.2006.2

[10] Weniger, K. and Zitterbart, M.: 'IPv6 autoconfiguration in large scale mobile ad-hoc networks', in Proceedings of the European Wireless, Florence, Italy, 2002, Vol. 1, pp. 142-148.

[11] Vaidya, N.H.: 'Weak duplicate address detection in mobile ad hoc networks', in Proceedings of ACM MobiHoc, Lausanne, Switzerland, 2002, pp. 206-216.

[12] García Villalba, L.J.; García Matesanz, J.; Sandoval Orozco, A.L.; Márquez Díaz, J.D. Auto-configuration Protocols in Mobile Ad Hoc Networks. Sensors 2011, 11, 4438-4461.

[13] García Villalba, L.J., García Matesanz, J., Sandoval Orozco, A.L., Márquez Díaz, J.D: Distributed Dynamic Host Configuration Protocol (D2HCP). Sensors 2011, 11, pp. 3652-3666.

[14] Clausen, T. and Jacket, P.:'Optimized Link State Routing Protocol (OLSR)', Internet Engineering Task Force, RFC 3626, 2003.

# E-D2HCP: Enhanced Distributed Dynamic Host Configuration Protocol

Luis Javier García Villalba[1, *], Julián García Matesanz[2], Ana Lucila Sandoval Orozco[1], Delfín Rupérez Cañas[1] and Tai-hoon Kim[3, 4]

[1] *Grupo de Análisis, Seguridad y Sistemas (GASS)*
*Departamento de Ingeniería del Software e Inteligencia Artificial (DISIA)*
*Facultad de Informática, Despacho 431, Universidad Complutense de Madrid (UCM)*
*Calle Profesor José García Santesmases s/n, Ciudad Universitaria, 28040 Madrid, Spain*
*E-mail: {javiergv, asandoval, delfinrc}@fdi.ucm.es*

[2] *Grupo de Análisis, Seguridad y Sistemas (GASS)*
*Sección Departamental de Sistemas Informáticos y Computación*
*- Lenguajes y Sistemas Informáticos y Ciencias de la Computación e Inteligencia Artificial -*
*Facultad de Ciencias Matemáticas, Despacho 310-F, Universidad Complutense de Madrid (UCM)*
*Plaza de Ciencias, 3, Ciudad Universitaria, 28040 Madrid, Spain*
*E-mail: julian@sip.ucm.es*

[3] *Department of Multimedia Engineering, Hannam University*
*133 Ojeong-dong, Daedeok-gu, Daejeon, Korea*
*E-mail: taihoonn@hannam.ac.kr*

[4] *Department of Information Technologies, Global Vision School Australia (GVSA)*
*20 Virgina Court, Sandy Bay, Tasmania, Australia*
*E-mail: taihoonn@gvsa.asia*

## Abstract

*Mobile Ad Hoc Networks (MANETs) consist of mobile nodes equipped with wireless devices. They do not need any kind of pre-existent infrastructure and are about self-managed networks. MANETs enable communication between mobile nodes without direct links and across multihop paths. To ensure correct operation of the routing protocols, Mobile Ad Hoc Networks, have to assign unique IP addresses to the MANET devices. Furthermore, the address assignment is an important issue when dealing with MANET networks because the traditional approaches are not applicable without some changes, having to provide new protocols for the address auto-configuration. These schemes must take into account the properties of MANETs such as dynamic topology, limited resources or lack of infrastructure. In this paper, we propose a stateful scheme for dynamic allocation of IP addresses in MANETs entitled E-D2HCP because it is based on a previous piece of work (D2CHP). This extension includes the network merging not covered by its predecessor. Simulation results show that the new protocol also improves D2HCP functionality in areas such as fault tolerance, concurrency and latency.*

**Keywords:** Auto-configuration, mobile ad hoc networks, MANETs, D2HCP, E-D2HCP

## 1. INTRODUCTION

A Mobile Ad Hoc Network (MANET) consists of a set of mobile nodes equipped with wireless devices. The nodes establish direct links with every node that is within its transmission range and thereby enable communication between nodes without direct links and across multihop paths. This kind of computer network is characteristic of its dynamic network topology, lack of any fixed infrastructure or administration, limited bandwidth and energy saving capacity. They are important features within the field of computer networks. One of the most important issues related to this type of network is the routing[1]. In order to provide routing in MANETs, it is necessary to design efficient mechanisms for address allocation and therefore the schemes for IP address allocation must meet certain requirements[2].

This work proposes a stateful auto-configuration protocol that guarantees the uniqueness of IP addresses under a wide variety of network conditions such as missing messages and merging and partitioning of networks. This work is structured into six sections; the first one is the present introduction. Section 2 shows the obligated references in the auto-configuration protocol scope of Mobile Ad Hoc Networks. Section 3 contains an itemized specification of the so-called Extended Distributed Dynamic Host Configuration Protocol (E-D2HCP), a proposal concerning IP addresses auto-configuration for MANETs. A

Published in IET Information Security
Received on 13th April 2010
Revised on 31st October 2010
doi: 10.1049/iet-ifs.2010.0085

ISSN 1751-8709

# Secure extension to the optimised link state routing protocol

*L.J. García Villalba*[1] *J. Garcia Matesanz*[2] *D. Rupérez Cañas*[1] *A.L. Sandoval Orozco*[1]

[1]*Departamento de Ingeniería del Software e Inteligencia Artificial, Grupo de Análisis, Seguridad y Sistemas (GASS), Universidad Complutense de Madrid*
[2]*Sección Departamental de Sistemas Informáticos y Computación, Grupo de Análisis, Seguridad y Sistemas (GASS), Universidad Complutense de Madrid*
*E-mail: javiergv@fdi.ucm.es*

**Abstract:** The design of routing protocols for mobile *ad hoc* networks rarely contemplates, in most cases, hostile environments. Consequently, it is common to add security extensions afterwards. One of the most important routing protocols is the optimised link state routing (OLSR), which in its specification assumes the trust of all nodes in the network, making it vulnerable to different kinds of attacks. This study presents an extension of OLSR, called COD-OLSR, which provides security for OLSR in the case of incorrect message generation attacks which can occur in two forms (identity spoofing and link spoofing). This is one of its main features, which takes into account the current topology of the node sending the message. The behaviour of COD-OLSR against different attackers in a variety of situations is evaluated. The simulation results show that COD-OLSR adds a slight overhead to OLSR and barely affects performance. The results also show that COD-OLSR is an interesting alternative to provide integrity in OLSR compared with classical mechanisms making use of cryptography, which is more complex and has a high overhead.

## 1 Introduction

Mobile *ad hoc* networks (MANETs) [1] are formed by mobile devices that can communicate with each other without appealing to a preexisting network infrastructure. Most routing protocols for these networks are designed without taking into account the possible malicious behaviour of any of the nodes, something that can be exploited to violate the security of the network. Optimised link state routing (OLSR) [2] belongs to this group of protocols where attackers can alter their behaviour, hence the need for an extension of this protocol. There are several techniques to provide integrity to the OLSR protocol. One of the most widespread is the use of digital signatures for authentication of the OLSR routing messages, of which there are two different variants or approaches: hop-by-hop and end-to-end.

Halfslund *et al.* [3] present a hop-by-hop approximation, where each node signs OLSR packets that are transmitted, discarding the signature of the previous node. This signature only verifies that the node which forwards the message is the same as the one that signs it, but it does not verify the authenticity of the original message. The implemented solutions use shared keys (symmetric) for signature creation and verification. However, they do not mention how the administration/exchange of keys or the initial authentication is carried out. They also propose a method to prevent replay attacks using timestamps, but the overhead increases significantly with this method.

Adjih *et al.* [4] propose schemes to authenticate OLSR messages in an end-to-end approach so that the nodes that receive OLSR messages can authenticate the node which generated the original message. However, replay attacks are still possible. To avoid such attacks another approach has been developed which is based on a distributed timestamp that can verify whether a message is obsolete or not.

Raffo *et al.* [5] design another end-to-end extension where the nodes send OLSR messages as well as an ADVanced Signature message (ADVSIG). Nodes that announce links sign the messages to authenticate the node that originated the message. Mechanisms are established which improve the protection of the protocol against external attacks and from other nodes, although this does significantly add to the overheads.

Raffo *et al.* [6] supply a method that includes the geographical position of the sender node in the control messages as well as an evaluation of the similarity of the links at the same time. This technique has the disadvantage of requiring specific hardware (GPS and directional antennas).

Papadimitratos and Haas [7] provide security in the routing of the link state by using asymmetric primitives. To do this, they assume that each node in the network has a pair of public/private keys, and diffusing in broadcast its public key certificate to the other nodes that are within $N$ hops. These broadcast messages can be periodic or not, depending on changes in the topology of the network, allowing a new node to find out the key to enter the area. Despite using distributed certification the overhead introduced is also important.

Finally, Nait-Abdesselam [8] presents a scheme for detecting and preventing 'Relay' attacks (Wormhole attacks). This technique is based on firstly an identification of the potential links in which the attack occurs, then in distinguishing the links where the Wormhole attack occurs from the ones that

whilst waiting for the node selected as server to complete its configuration process.

- Even if any node does not exist in *WaitingAllocation* state, but nodes in *Suspended* state exist, the client node also goes to *Suspended* state. It means that either the neighbour node has found another one that is performing the configuration of its interface, or a chain of nodes exists in *Suspended* state from neighbour node, until a node that has already begun the process. For this reason, the client node suspends the process of address configuration from network interface for a period of time, so that the neighbour nodes can end their owner process. When the waiting time expires, the client node restarts the process by going to *SearchingServer* state.
- If the answers come from nodes in *SearchingServer* state, it means that the message sender has not found any node that began or ended the configuration process and this node has initiated the process of creating a new network and it sends a *ClientSynchronization* message and Synchronizing is passed on. In this state, the nodes which have priority are identified.
- Whether *nRetr* counter gets *MAX_DISC_RETRIES* value and the node does not receive any *Server_Offer* message, it is assumed that a network exists in the environment, thus the client node initiates the generation of a new network.

If the client node has received *Server_Offer* messages, analyzes the status of the nodes that have responded and verifies that there are nodes in *Configured* state:

- It analyzes the networks that have responded checking the *NetworkID* of the server node and it selects a network to join.
- It analyzes the nodes that responded from the chosen network and chooses one as "server node" with priority going to those nodes that dispose free addresses, have an owner address range and do not need to perform a remote request.
- After selecting a server node, the notification of its selection sends a *Server_Poll* message to it and therefore the client node passes it to *WaitingAllocation* state while it waits to receive an *IP_Assigned* message during an *ADDR_ALL_ TIME* period of time. If after this period of time the client node has not received any answer the nRetr counter value is increased from *WaitingAllocation* state. If this counter value is lower than *MAX_ALL_RETRIES*, *Server_Poll* message is then forwarded and the same state is maintained. In the case of getting *MAX_ALL_RETRIES* value it is passed on *SearchingServer* state and the process is restarted. If the node selected as server offers an address range that they are not its own, it sends the *IP_Range_Request* message to owner node of range ("remote node") requesting an offered address range. The "remote node" reply to server node with the *IP_Range_Return* message delivering requested range.
- The server node sends the *IP_Assigned* message with which it confers the administration of an address range to client node and it provides an address to set the network interface identified by hardware address, Therefore the client node passes onto *Configured* state.

## 3.2. NETWORK INITIALIZATION

The network initialization can be performed by a single node or group of nodes. If a client node does not find any neighbour, it chooses an address range and generates a *NetworkId*. The node configures the network interface with this information. If the number of nodes is bigger than one in the initialization process, the group of nodes set their state to Synchronizing. Then, the nodes select a subgroup, called the network precursors. These precursors choose an address range and *NetworkId*. Following this, the precursor nodes distribute the address to the other nodes.

## 3.3. SYNCHRONIZATION

Nodes of a MANET synchronize from time to time to keep track of IP address assignment in the entire network and to detect any IP address leaks. In the original protocol, the synchronization process involved every node broadcasting its pool of addresses. In E-D2HCP, the protocol monitors the changes made in routing table.

OLSR send messages periodically with network topology information. E-D2HCP updates the address block table with this information. The address block table maintained by E-D2HCP stores the following information:

*Owner address*: node address.

*First address*: the first address of the block from node with address "*Owner Address*".

*Last address*: the last address of the block from node with address "*Owner Address*".

*Lost flag*: this flag indicates the existence of a route to this node.

When OLSR report the discovery of a nonexistent node in the address block table, it creates a new entry in the table. The new address is assigned by the node that has a block with that address. A search is made for the owner node in the table and then, the block is updated. There are two different cases:

- The new address is less than the owner address: The new first address in the owner block is the owner address. The new block will be [*new address, owner first address, owner address minus one, false*].
- The new address is greater than the owner address: The new last address in the owner block is the new address minus one. The new block will be [*new address, new address, and owner last address, false*].

When OLSR loses a node, E-D2HCP examines its entry. If the lost flag from this entry is true, the entry is deleted. In this case, the address block assigned to node A is handed over to node B. This node B has the preceeding address block to the block of node A. If there is no preceeding block, the node B will be the node with the next block. If the lost flag value is false, then it is set to true. This technique prevents OLSR failures, and route changes.

### 3.4. PROTOCOL MESSAGES

Message header sending by E-D2HCP protocol has the following fields:
*Type*: The kind of contained message. It can have the following values: 1 (*Server_Discovery*), 2 (*Server_Offer*), 3 (*Server_Poll*), 4 (*IP_Assigned*), 5 *(IP_Range_Request*), 6 (*IP_Range_Return*).
*Reserved*: Reserved bits for future usage. It contains the value 0.
*Length*: The message length.
Then the different messages utilized in the protocol are detailed.

- ***Server_Discovery.*** This message is sent by a node which wishes to set a network interface. The fields are the following:
  *Hwtype*: The type of hardware, for example: Ethernet (1) or IEEE 802 Networks (6). To refer STD 2 - Assigned number of Internet for a completed list.
  *Hwlength*: Hardware address length in bytes. Ethernet and token-ring use 6, for example.
  *G*: Flag which indicates whether the client node disposes of a link to another network.
  *N*: This field indicates the type of request. The value 0 refers that it does not exist any constraint for networks. The value 1 denotes that a network with a determinate group of identifiers is being looked for. The value 2 indicates it is looking for a network created by a device with a determinate hardware address.
  *I*: If it contains the value 1, this indicates that a network with Internet access is looked for.
  *S*: Required service number by the network (DNS).
  *It*: Current iteration of server discovery process.
  *Client Hardware Address*: Hardware address of network interface that wishes to configure.
  *Gateway Address*: Optional field. This field is taken into account if the flag G is activated.
  *Network id*: Reference identifier when the value of field N is 1 or 2.
  *Serv id i*: Optional field. Each field contains the identifier of a service. There will be as many fields as the value of field S indicates.
- ***Server_Offer.*** This message is sent in response to *Server_Discovery* message. This message may be sent as nodes which have ended the configuration process or as nodes which have not completed the process yet. The client node is notified of the node state and, if it is configured, relative information is sent to network. The fields are the following:
  *Hwtype*: The kind of hardware, for example: Ethernet (1) or IEEE 802 Networks (6). To refer STD 2 - Assigned number of Internet for a completed list.
  *Hwlength*: Hardware address length in bytes. Ethernet and token-ring use 6, for example.
  *S*: It indicates the node state which has replied to the *Server_Discovery* message.
  *NG*: This field contains the gateway number that such a network has.
  *access*: It indicates the access constraints to network.
  *Client Hardware Address*: Hardware address of network interface that wishes to configure the client node.
  *Server Address*: Server IP address.
  *Network id*: Network identifier.
- ***Server_Poll.*** This message is sent by a client node to server node. Once a client node has received a *Server_Offer* message from one or more nodes, it sends a *Server_Poll* message to one of them. This message indicates to the recipient that it has been selected by the client node as the server. The fields are the following:
  *Hwtype*: The kind of hardware, for example: Ethernet (1) or IEEE 802 Networks (6). To refer STD 2 - Assigned number of Internet for a completed list.
  *Hwlength*: Hardware address length in bytes. Ethernet and token-ring use 6, for example.
  *Reserved*: Reserved bits for future usage. It contains the value 0.
  *Client Hardware Address*: Hardware address of network interface that wishes to configure the client node.
- ***IP_Assigned.*** Message sent to client node by a selected server node. This message provides the management with an address range and an address to configure the network interface identified by a hardware address. The fields are the following:
  *Hwtype*: The kind of hardware, for example: Ethernet (1) or IEEE 802 Networks (6). To refer STD 2 - Assigned number of Internet for a completed list.
  *Hwlength*: Hardware address length in bytes. Ethernet and token-ring use 6, for example.
  *NG*: Available Gateway number in the network.
  *Reserved*: Reserved bits for future usage. It contains the value 0.
  *Client Hardware Address*: Hardware address of network interface that wishes to configure the client node.
  *First IP Address*: First address of granted address range.
  *Last IP Address*: Last address of granted address range.
  *Client IP Address*: Assigned address to network interface.
  *Network mask*: Network mask.
  *Network id*: Network identifier.
  *Gateway i Address*: Address of gateway number *i*.
  *Network i Address*: Network address whose gateway *i* gives access to it.
- ***IP_Range_Request.*** Message sent to owner node of range ("remote node") requesting an offered address range. The fields are the following:
  *Server Address*: Address of selected node as server. It is the node which initiates the request for an address range to a remote server.
  *Remote Server Address*: Address of the remote server which an address range is requested from.
- ***IP_Range_Return.*** When a configured node receives the request of an address range from another node in the

network, in the case of disposing of free addresses, it replies with an *IP_Range_Return* message.
The fields are the following:
*Server Address*: Address of selected node as server. It is the node which initiates the request of an address range to a remote server.
*Remote Server Address*: Address of the remote server which an address range is requested from it.
*First IP Address*: First address from address range.
*Last IP Address*: Last address from address range.
*Client IP Address*: IP address granted to network interface.
*Network Mask*: Network mask.

- ***ClientSynchronization.*** When two adjacent nodes start the auto-configuration process and there is no other node that has progressed in this task, they send a synchronization message to set which of them will have priority to initiate the network. The fields are the following:
  *Hwtype*: The kind of hardware, for example: Ethernet (1) or IEEE 802 Networks (6). To refer STD 2 - Assigned number of Internet for a completed list.
  *Hwlength*: Hardware address length in bytes. Ethernet and token-ring use 6, for example.
  *Reserved*: Reserved bits for future usage. It contains the value 0.
  *Client Hardware Address*: Hardware address of network interface that wishes to configure the client node.
  *sHwtype*: Type of hardware, for example, Ethernet (1) or Networks IEEE 802 (6). With reference to STD 2 – Assigned numbers from the internet for a complete list.
  *sHwlength*: Hardware address length in bytes. Ethernet and token-ring use 6, for example.
  *Reserved*: Reserved bits for future usage. It contains the value 0.
  *Server Hardware Address*: Hardware address of network interface that wishes to realize the synchronization.
  *Delay*: It indicates the time in milliseconds that it has elapsed from when the sender node started the configuration process until it received the *Server_Offer* message from the other node.
- ***Hello.*** Message periodically sent to detect possible network fusions. The fields are the following:
  *Node Address*: IP address of sender node.
  *Network Id*: Network identifier.

## 4. E-D2HCP versus D2HCP

E-D2HCP is an extension of D2HCP. The following changes are made to the protocol D2HCP:
Two new messages have been added:

- ***ClientSynchronization***: When two adjacent nodes begin the auto-configuration process and another node that has progressed on this task, doesn't exists, they send a synchronization message to establish which of them will have priority to initiate the network.
- ***Hello***: Message sent periodically to detect possible network merging.

The following states have been modified:

- ***SearchingServer***: State indicating the node has sent the Server_Discovery message
- ***WaitingAllocation***: It enters this state when the client node has found a server that can provide a valid address.
- ***Suspended***: A client node enters this state when none of the nodes that have responded to the *Server_Discovery* message have completed the configuration process
- ***Synchronizing***: When two adjacent nodes have not found any node that has started or ended the configuration setup, they go to start a new network. In this state the decision is made as to which nodes have priority in this process.

## 5. SIMULATION AND RESULTS

For the performance evaluation of auto-configuration protocol E-D2HCP we have taken into account the metrics of latency in assignment, control message number emitted in each address configuration and overhead as well as packet number in Bytes.

### 5.1. PROTOCOL MESSAGES

E-D2HCP was implemented and evaluated with Network Simulator 3. In this section, we analyze the impact of the number of nodes in the protocol overhead and latency. Table 1 summarizes the main parameters used during simulations.

**Table 1 – Simulation Parameters**

| Parameter | Values |
|---|---|
| Simulation Area | 750x750m$^2$ |
| Routing protocols | OLSR |
| OLSR update/hello interval | 5s/2s |
| Node density | 40 nodes/km$^2$ |
| Simulation time | 600s |
| Mobility model | Random Waypoint |
| MAC Protocol | 802.11 |
| Transmission range | 276m |
| Network Address Class | B and C |

In Figures 1 and 2, we verified the scalability of E-D2HCP. We have evaluated the latency and overhead in the address assignment process.
Figure 1 shows the evolution of the latency value depending on the node number in the network, showing values of IPv4 addresses from Class C, observing that the E-D2HCP latency decreased by 36% respect to D2HCP.
Figure 2 shows the message number exchanged during IP address configuration process. This one is reduced substantially in E-D2HCP according to the node number increases in the network. With the improvements in E-D2HCP a 21% reduction

in overhead from 150 nodes with respect to overhead presented in D2HCP is achieved.
The protocol performance is worse when the percentage of assigned addresses increases. However, when the network consists of 240 nodes (95 %), the latency time is less than one second (see Figure 1).
In Figure 3 we analyzed the latency for two types of IPv4 address classes: Class B and Class C. We see that the latency for the assignment of class C address grows quicker than class B addresses. It is observed from 75 nodes.
These results indicate that the parameter which determines the latency, in general, is the occupied address percentage.
In the case of class C addresses, by having fewer addresses than class B, it cannot carry out a local assignment being necessary to request the address to another node, increasing the necessary time to complete the process.
In general, the average latency in address assignment process is low for both address classes in this scenario.
In figure 4 we show the average number of emitted control messages in each address configuration.
The E-D2HCP protocol presents a large reduction in the control packet overhead in the network.
In fact, in most cases, the configuration is realized in a local way, i.e., the neighbour will assign the address to a new node. In the occasion where it couldn't assign addresses locally, a unicast transmission with chosen server is realised.
The message average number exchanged is very similar for both IPv4 address classes. From 60 nodes class C has a slight increase in the average number of messages with respect to class B.
Finally, in figures 5 and 6 we show the overhead in the packet and Bytes number.
Figure 5 shows that the overhead in packet number is very similar up to 75 nodes for both address classes. From 75 nodes it increases the received packets without appreciate difference in the two classes of analyzed networks.
In figure 6 we have the overhead in number of Bytes. The graph shows a similar pattern for both classes except at 100 nodes where the received bytes are double for class B than for class C because there's more possibility of addressing.

## 6. CONCLUSIONS AND FUTURE WORK

The E-D2HCP protocol is a stateful approach and therefore it does not require Duplicate Address Detection (DAD), which reduces the overheads. Additionally, monitoring the routing protocol simplifies the synchronization process. The latency depends on the address space size. If the percentage of allocated addresses is small, the chances of finding a server with available addresses increase. In this case, the latency is small.
E-D2HCP is an extension of D2HCP protocol. Among the improvements which it contemplates, the following exist:

- Network Merging.
- Support for the failure of client-server links during the configuration process.
- Improvements in the network initiation process achieving node synchronization which they wish to initiate the network concurrently.
- Improvement in latency.

Topics for future research include the development of QoS and security extensions.

## ACKNOWLEDGEMENTS

This work was supported by the Ministerio de Industria, Turismo y Comercio (MITyC, Spain) through the Project Avanza Competitividad I+D+I TSI-020100-2011-165. This work was also supported by the Security Engineering Research Center, granted by the Ministry of Knowledge Economy (MKE, Korea).

## REFERENCES

**1.** S. Corson and J. Macker, Mobile Ad Hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations. Internet Engineering Task Force, RFC 2501 **(1999)**.
**2.** E. Bacelli, Address Autoconfiguration for MANET: Terminology and Problem Statement. Internet Engineering Task Force, Internet Draft draft-ietf-autoconfstatement-04 **(2008)**.
**3.** The NS-3 Network Simulator. Available online: http://www.nsnam.org.
**4.** R. Droms, Dynamic Host Configuration Protocol. Internet Engineering Task Force, RFC 2131 **(1997)**.
**5.** S. Nesargi and R. Prakash, MANETconf: Configuration of Hosts in a Mobile Ad Hoc Network. Proceedings of the IEEE INFOCOM, New York, USA, pp. 1059-1068 **(2002)**.
**6.** S. Cheshire, B. Aboba and E. Guttman, Dynamic Configuration of IPv4 Link-Local Addresses. Internet Engineering Task Force, RFC 3927 **(2005)**.
**7.** S. Thomson, T. Narten, and T. Tinmei, IPv6 Stateless Address Autoconfiguration. Internet Engineering Task Force, RFC 4862 **(2007)**.
**8.** C. Perkins, J. T. Malinen, R. Wakikawa, IP Address Autoconfiguration for Ad Hoc Networks. Internet Engineering Task Force, Internet Draft draft-ietfmanetautoconf-01 **(2001)**.
**9.** K. Weniger andM. Zitterbart, IPv6 Autoconfiguration in Large Scale Mobile Ad-Hoc Networks. Proceedings of the European Wireless, Florence, Italy, 1, pp. 142-148 **(2002)**.
**10.** N. H. Vaidya, Weak Duplicate Address Detection in Mobile Ad Hoc Networks. Proceedings of ACM MobiHoc, Lausanne, Switzerland, pp. 206-216 **(2002)**.
**11.** M. Mohsin and R. Prakash, IP Address Assignment in a Mobile Ad Hoc Network. Proceedings of the 2002 Military

Communications (MILCOM 2002), Anaheim, CA, USA, Vol. 2, pp. 856-861 **(2002)**.

**12.** M. Thoppian, and R. Prakash, A Distributed Protocol for Dynamic Address Assignment in Mobile Ad Hoc Networks. IEEE Transactions on Mobile Computing, 5 (1), pp. 4-19 **(2006)**.

**13.** C. Bernardos, M. Calderon, H. Moustafa, Ad-Hoc IP Autoconfiguration Solution Space Analysis. Internet Draft **(2008)**.

**14.** C. Bernardos, M. Calderon, H. Moustafa, Survey of IP Address Autoconfiguration Mechanisms for MANETs. Internet Draft **(2008)**.

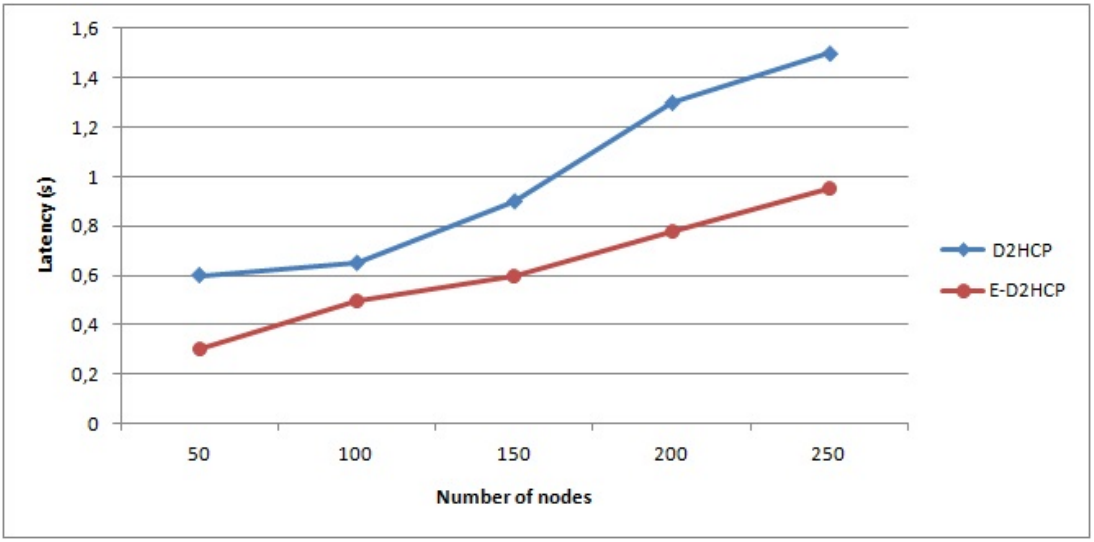
**15.** C. Bernardos, M. Calderon, H. Moustafa, Evaluation Considerations for IP Autoconfiguration Mechanisms in MANETs. Internet Draft **(2008)**.

**16.** L. J. García Villalba, J. García Matesanz, A. L. Sandoval Orozco and J. D. Márquez Díaz, Auto-configuration Protocols in Mobile Ad Hoc Networks. Sensors, 11, pp. 3652-3666 **(2011)**.

**17.** L. J. García Villalba, J. García Matesanz, A. L. Sandoval Orozco and J. D. Márquez Díaz, Distributed Dynamic Host Configuration Protocol (D2HCP). Sensors, 11, pp. 4438-4461 **(2011)**.

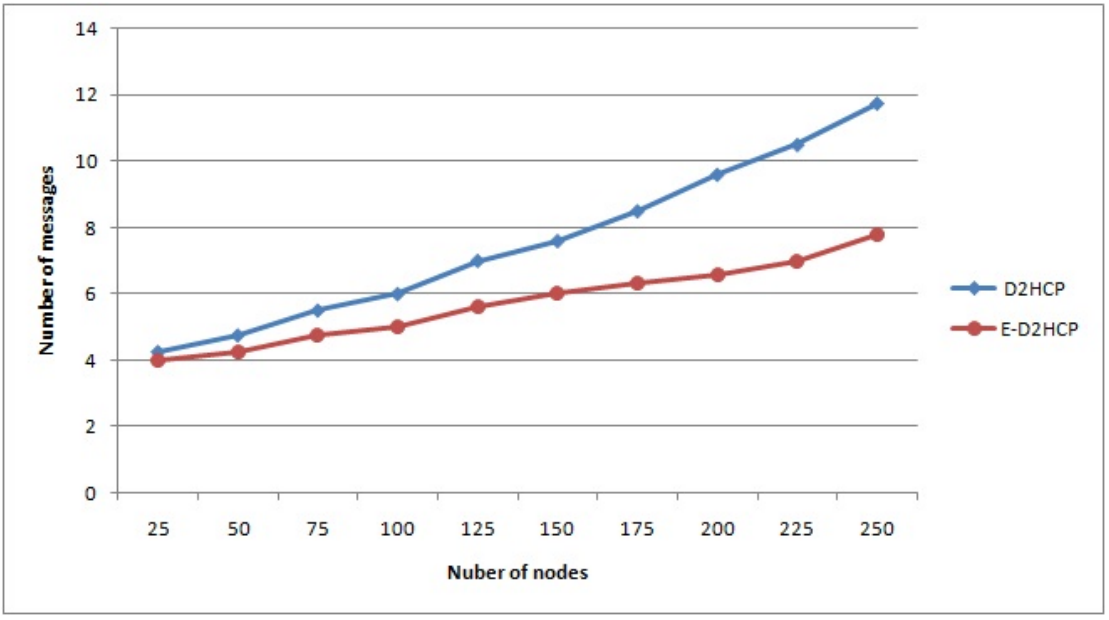
**18.** T. Clausen and P. Jacket, Optimized Link State Routing Protocol (OLSR). Internet Engineering Task Force, RFC 3626 **(2003)**.

**Fig. 1. Latency graph.**



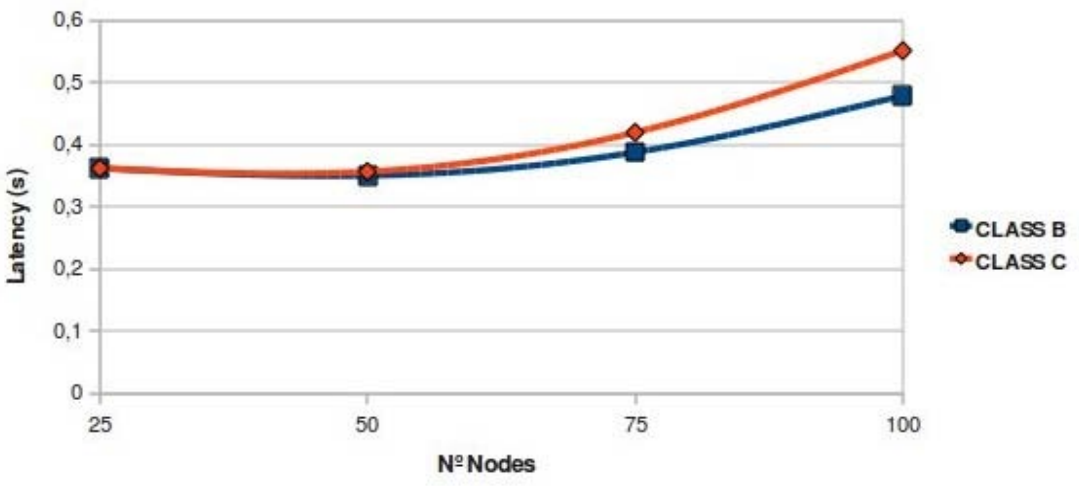

**Luis Javier García, E-D2HCP: Enhanced Distributed Dynamic Host Configuration Protocol, Fig. 1**

**Fig. 2. Overhead graph.**



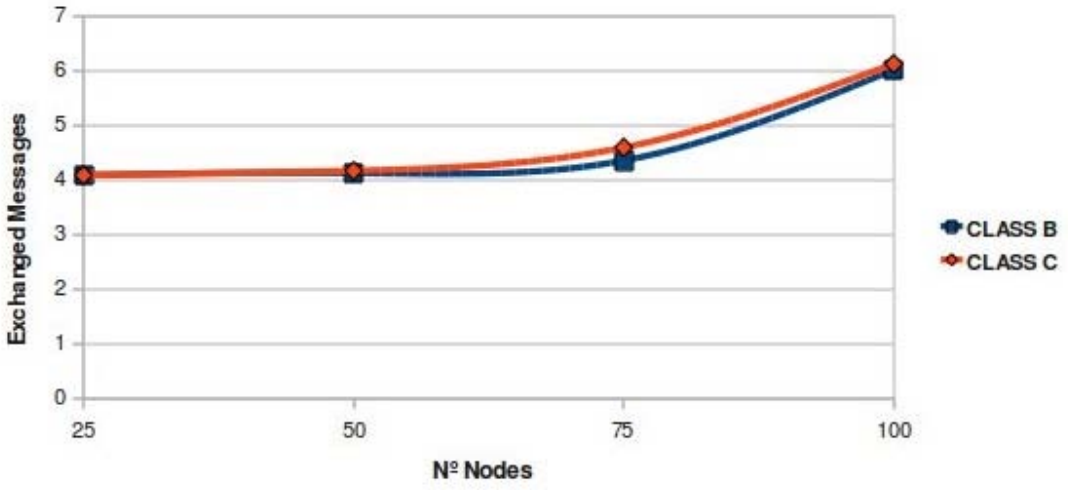

**Luis Javier García, E-D2HCP: Enhanced Distributed Dynamic Host Configuration Protocol, Fig. 2**

**Fig. 3. Latency comparative.**



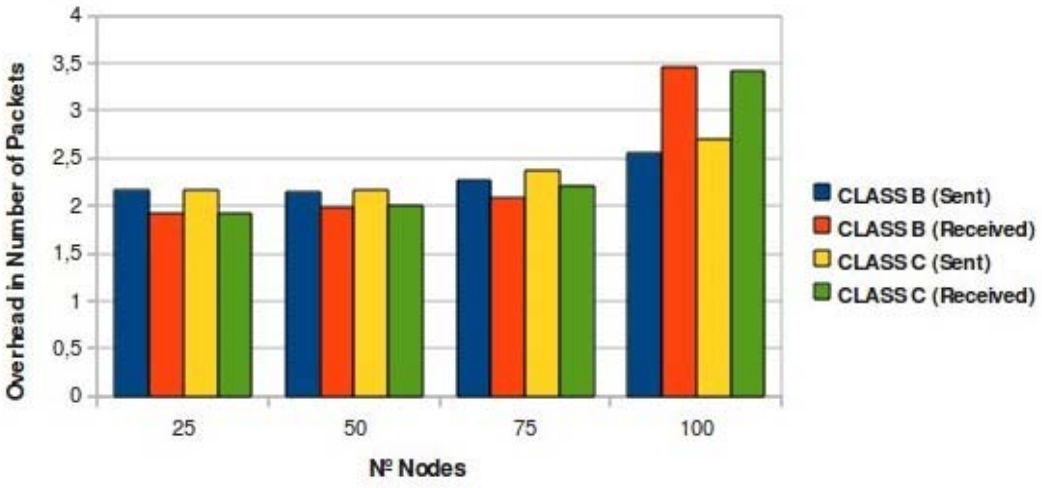

**Luis Javier García, E-D2HCP: Enhanced Distributed Dynamic Host Configuration Protocol, Fig. 3**

**Fig. 4. Exchanged Messages.**



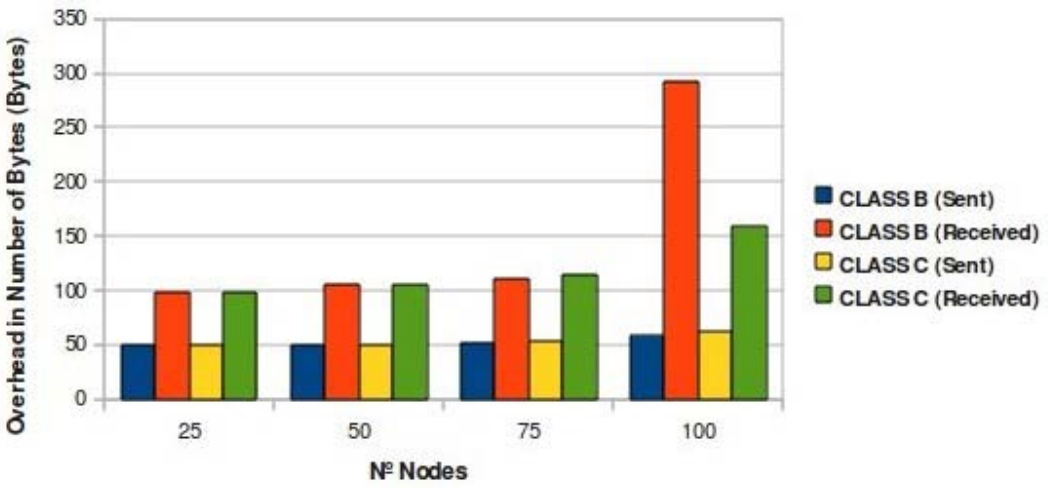

**Luis Javier García, E-D2HCP: Enhanced Distributed Dynamic Host Configuration Protocol, Fig. 4**

**Fig. 5. Overhead in Number of Packets.**




**Luis Javier García, E-D2HCP: Enhanced Distributed Dynamic Host Configuration Protocol, Fig. 5**

**Fig. 6. Overhead in Number of Bytes.**




**Luis Javier García, E-D2HCP: Enhanced Distributed Dynamic Host Configuration Protocol, Fig. 6**

# Secure Auto-Configuration Protocols in Mobile Ad Hoc Networks

Ana Lucila Sandoval Orozco[1], Julián García Matesanz[2], Luis Javier García Villalba[1,*], José Duván Márquez Díaz[3] and Tai-hoon Kim[4, 5]

*[1] Grupo de Análisis, Seguridad y Sistemas (GASS)*
*Departamento de Ingeniería del Software e Inteligencia Artificial (DISIA)*
*Facultad de Informática, Despacho 431, Universidad Complutense de Madrid (UCM)*
*Calle Profesor José García Santesmases s/n, Ciudad Universitaria, 28040 Madrid, Spain*
*E-mail: {asandoval, javiergv}@fdi.ucm.es*

*[2] Grupo de Análisis, Seguridad y Sistemas (GASS)*
*Sección Departamental de Sistemas Informáticos y Computación*
*- Lenguajes y Sistemas Informáticos y Ciencias de la Computación e Inteligencia Artificial -*
*Facultad de Ciencias Matemáticas, Despacho 310-F, Universidad Complutense de Madrid (UCM)*
*Plaza de Ciencias, 3, Ciudad Universitaria, 28040 Madrid, Spain*
*E-mail: julian@sip.ucm.es*

*[3] Grupo de Redes de Computadores e Ingeniería de Software (GRECIS)*
*Departamento de Ingeniería de Sistemas, Universidad del Norte*
*Km. 5 Autopista a Puerto Colombia, Barranquilla, Colombia*
*E-mail: jmarquez@uninorte.edu.co*

*[4] Department of Multimedia Engineering, Hannam University*
*133 Ojeong-dong, Daedeok-gu, Daejeon, Korea*
*E-mail: taihoonn@hannam.ac.kr*

*[5] Department of Information Technology, Global Vision School Australia (GVSA)*
*20 Virgina Court, Sandy Bay, Tasmania, Australia*
*E-mail: taihoonn@gvsa.asia*

## Abstract

*Ad hoc networks are built on the basis of a communication without infrastructure and major investigations have focused on the routing problems and auto-configuration. However, there is little progress in solving the secure auto-configuration problems in mobile ad hoc networks (MANETs), which has led to the proliferation of threats given the vulnerabilities of MANETs. It is clear that ad hoc networks have no centralized mechanism for defense against threats, such as a firewall, an intrusion detection system or a proxy. Therefore, it is necessary that the defense of interests of each of the ad hoc components is the responsibility of each member node. This paper shows the most common threats to ad hoc networks and reviews several proposals that attempt to minimize some of these threats, showing their protection ability and vulnerabilities in light of the threats that might arise.*

**Keywords:** MANETs, mobile ad hoc networks, secure auto-configuration, threats.

## 1. INTRODUCTION

MANET technology is used to immediately provide secure access between multiple mobile nodes without the need for a pre-set communications infrastructure achieving a multi-hop architecture. These networks are identified by two basic principles: routing and auto-configuration.

While there is already quite a lot of established work undertaken on routing[1-4] and consequently those related to secure routing[5-8], there is still room for continuous improvements on those which are still under construction, notably those related to auto-configuration and in particular, those in connection with secure MANET auto-configuration. Thus, this paper shows the most important works carried out concerning the latter.

Insertion of a node to the MANET involves implementing initial configuration mechanisms[9-10], such as assigning an available IP address before this node can participate actively in

the network. There are three types of solutions to carry out this assignment: *stateful*, *stateless* or *hybrid*.
In *stateful* solutions, addresses are assigned by the network, therefore the network should maintain the status information of addresses that have been assigned and/or released.
In *stateless* solutions, the addresses are assigned by the same node that enters the MANET. This node should run a test for duplicate address detection (DAD) in order to determine the uniqueness of the assigned address.
The *hybrid* solutions combine aspects of both previous types of solutions to improve the scalability and reliability of auto-configuration mechanisms.
All proposals have advantages and disadvantages in terms of solving the following problems: uniqueness of addresses, network initialization, node departure, network partitioning and network merging. However, all lack a mechanism to ensure the authenticity of the address owner at the time in which the auto-configuration is carried out. As a result, a malicious node can spoof any node already set up to hijack its traffic; preventing other nodes entering the network, sending messages with false addresses; denial of service by flooding the network with unsolicited messages from fake addresses, rejecting the possibility that other nodes can access the network or the refusal to accept the insertion of a new node, when the auto-configuration mechanism requires that all nodes confirm the entry of a new member to the MANET.
Although studies over the authenticity of the nodes entering the MANET during auto-configuration have been minimal, the aim of this paper is to show how they have presented some solutions to this problem and show some of its shortcomings from the perspective of the characteristics to be evaluated for potential threats within the auto-configuration process.
This piece of work, including the introduction, is organized into four sections as follows. Section 2 shows an overview of possible threats that may occur within a MANET. In Section 3, the highlights of some proposed solutions to secure MANET auto-configuration are reviewed and analyzed. Finally, conclusions are presented in Section 4.

## 2. THREATS IN AUTO-CONFIGURATION

In the processes applied during the execution of the mechanisms of auto-configuration, predictable and reliable behaviour from the nodes that compose the MANET is expected, as much from those which enters as from those already inside. However, this is not always the case, and malicious nodes can potentially be causing some damage, such as interference of messages, node impersonation, denial of service, spoofing and eavesdropping among others.
In this paper we use the classification proposed for Wang et al[11] and Buiati et al[12] to specify the security threats:

- **Address Spoofing Threat**: A malicious node may deliberately choose an assigned or free IP address for their attack. In the first case, the malicious node teases to any configured node as its victim and hijacks his traffic, and in the second one, the node assigns the free IP address to itself to participate in the network, gathering important information necessary to execute active attacks, such as denial of service.
- **Address Space Exhaustion** Threat: A malicious node can claim as many IP addresses as possible until exhausting the address space. This node may request the assignment of addresses to a ghost node (fake nodes). This way the malicious node could prevent other nodes from being configured and entering into the MANET.
- **Address Conflict Threat**: A malicious node can assign a duplicate address to a requester from a possible set of addresses already in use. Thus, it will create, in the DAD process, a *Black Hole* attack for *Address Reply* Messages (AREP) and lead to a address conflict in the MANET.
- **False Address Conflict Threat**: A malicious node might answer in an unscrupulous way, during the DAD process, messages *AREQ* (*Address Request*) with false addresses in messages *AREP* (*Address Reply*) that cause conflict with the requester node. Since the victim nodes cannot verify the authenticity of the proposed address, it would have to give up their address and find a new one. The malicious node may change its IP address to execute its attack.
- **Denial of Service** Threat: A malicious node could, in an auto-configuration process, act as a requester and send *AREQ* messages to multiple initiator nodes simultaneously. Similarly, a malicious node may send many fake DAD messages, causing an overload of traffic.
- **Sybil (Multiples Identities) Threat**: A node illegally claims multiple identities (*Sybil node*). This node can build a new identity or steal an existing legal node. In general, a Sybil node could demand or assign itself many IP addresses.
- **Negative Reply Threat**: When assigning a new IP address the approval of all pre-configured nodes is required, an attacker can send a negative response to avoid the entry of the new node.

## 3. SECURE AUTO-CONFIGURATION

The following are currently the most significant proposals that include secure IP address auto-configuration. The operation of each protocol and what threats they are capable of preventing are explained.
Wang et al[11] propose a scheme of secure IP address auto-configuration for MANETs, which binds each IP address with a public key allowing each node to authenticate itself into the network and thus prevent spoofing identity and other attacks.
The following are considered as the four main security threats surrounding MANET auto-configuration: *Address Spoofing*, *False Address Conflict*, *Address Space Exhaustion* and *Negative Reply* threats.
These threats try to be avoided by carrying out an authentication of identity and this paper proposes to relate every IP address to

a public key by means of an one-way hash operation, therefore the owner node of an IP address must use the correspondent key public in order to be authenticated by the network of a unilateral way

It initiates from the following assumption: The MANET is a network with completely private IP addresses. Therefore, all 32 bits (IPv4) or 131 bits (IPv6) can be used to address nodes in the MANET.

In general, in the proposed scheme, Node A, which wants to participate in an existing MANET or start a new one, must first randomly generate a key pair (one public and one private) and one secret key. In the second instance, node A calculates a hash of 32 bits for IPv4 or 131 bits for IPv6.

After calculating the hash value, the node in question temporarily uses the resulting value as its IP address, starts a timer and broadcasts a *Duplicate Address Probe* message (*DAP*)[13] used to check duplicated addresses on the network.

If a node (Node B) configured within the MANET where Node A wants to enter, finds that the IP address contained in the *DAP* message issued by node A is equal to it, then it must verify the authenticity of the *DAP* message. First, node B must check that the IP is equal to the resulting hash of the received public key. Secondly, Node B verifies the signature of Node A, if it finds that such a signature is correct, then Node B checks if public key of node A is equal to it, and finally verifies the decryption function. If at least one of last two checks is not fulfilled, it can be confirmed that there has been an address spoofing attack and therefore Node B sends an *Address Conflict Notice* message (*ACN*) via broadcast, and discards the received *DAP* message.

Node A, in turn, waits as long as configured in an internal timer. If it does not receive an *ACN* message, it assumes that the IP is not in use and permanently assigns the address. If instead, it receives an *ACN* message from some node, before starting the process again, it must verify the authenticity of the *ACN* message received and the signature of the node issuing the *ACN*. If these checks are correct, Node A is safe that the IP address is assigned to another node and must start the procedure to generate a new pair of public/private keys and secret key; otherwise Node A simply discards the *ACN* message and thus prevents *False Address Conflict* and *Negative Reply Attacks*.

It is clear that the proposed methodology in the auto-configuration process forces a potential attacker to find, before launching an attack, the public key for which the hash function result is equal to the IP address of the victim, since the controls in the nodes include verification of the identity of the sender node. This process must be applied for each message sent, however the protocol clearly controls address conflict, negative reply and address spoofing attacks but does not counteract the address exhaustion attack since it does not have a way to specify what node is given what IP addresses, allowing one node to repeat the process as often as desired. This process should be subject to an *ACN* message which certifies that the node will repeat the process because of IP address duplication.

Buiati et al[12] propose a secure model for auto-configuration in MANET, based on a distributed and self-organizing certificate system, and also include intrusion detection techniques to improve its safety. The proposed model is built on the protocol DCDP[14] with the improvements proposed for Mohsin et al[15], adopting a collaborative trust model described as "K-out-of-N". So, when a new node wants to enter the network, it must earn the trust of K of the N total nodes in the network in order to be accepted into it. To this end, nodes are able to generate certificates with varying degrees of trust. Thus, a distrusted node in the network can't attack by requesting multiple IP addresses to exhaust them or respond to configuration requests in a malicious manner, as well as allow the implementation of intrusion detection techniques[16].

For the security model, an adversary is defined as any node that produces messages with incorrect auto-configuration protocol information. It then specifies that an adversary can attack the network in two ways: request attacks, where the adversary creates a great number of anomalous messages requesting auto-configuration services, or server attack, where the attacker responds maliciously to requests made by other nodes in the network. In order to avoid these types of attacks, the authors differentiate between trusted and distrusted nodes, avoiding the participation of the latter in the auto-configuration protocol.

Even so, there is the possibility that a trusted node is compromised, so the ability to detect reliable nodes that begin to behave abnormally must be implemented as well. This means that the auto-configuration protocol messages must be authenticated so that an adversary cannot create messages on behalf of another node in the network, being capable of detecting and accusing the adversary nodes. In addition, this detection and accusation system should be implemented collaboratively to prevent an adversary of accusing correct nodes of the network, using the same model "K-out-of-N" explained above.

Authentication of auto-configuration protocol messages is performed using digital signatures, which are built based on digital certificates generated by a distributed certifying authority. This is where the model "K-out-of-N" is applied directly, since, even though every one of the nodes can perform the functions of certifying entity, the entity's private key is split between any subset of K nodes in the MANET. When a new node (one that has not been previously connected to the network) wants to get a digital certificate to identify itself to the MANET, it must take a temporary IP address to request a digital certificate to his 1-hop neighbours. When the MANET nodes receive this request, they can issue a partly signed certificate, depending on the policies established, and send it to the requesting node. After receiving K different certificates, the new node has the ability to build a full certificate and begin the auto-configuration process, discarding the temporary IP. The use of a temporary IP can cause collision problems if the IP is already in use in the network, but it is proposed to use a range of dedicated IPs for this purpose.

The biggest problem in the proposed model is the value of K. A high K value increases security, but reduces the availability of the system because members are less likely to find enough

nodes to retrieve the necessary key to the CA. Conversely, if K is small, the availability of the auto-configuration service increases, but the system becomes more vulnerable to attacks by adversaries.

Cavalli et al[17] propose a secure auto-configuration protocol adapted to the performance of *ad hoc* networks, which includes the authentication of the nodes within the network that they will be participating in. In general, it is intended to satisfy the following items with their secure auto-configuration protocol:

- At any time a node must be able to enter or leave the network quickly. Likewise, the network must be able to securely and quickly deliver an IP address to a new node. On the other hand, the abrupt departure of a node must not cause chaos within the network.
- To avoid duplicate IP address conflicts, the protocol must ensure that under no circumstances a node enters the network with its own IP address, but instead the network must be able to deliver the right address to join the MANET.
- The protocol should allow each node to check the veracity of the members of the network to which they belong.
- The protocol should be extremely careful with denial of service. For example, it must not allow a malicious node from monopolizing all IP addresses on the network.

The protocol, in addition to satisfying the described requirements above, wants to meet two broad objectives, the first is to provide a mechanism for IP address auto-configuration for nodes belonging to an ad hoc mobile network, optimizing resources such as bandwidth and time, and the second objective is to allow public key exchange between nodes within the network to ensure the authentication.

The proposed protocol ensures safe IP address auto-configuration including the management of public keys for authentication, which allows avoiding the spoofing attack. However one of its greatest failings is that it neither provides nor supports merging networks or prevents malicious behavior of network participants after these have been authenticated, among these the denial of service attack is worth mentioning since, for example a malicious node can authenticate n successive times with n different identities in order to exhaust the available addresses; and another form of attack is that the malicious node refuses to authenticate incoming nodes.

According to Hu et al[18] the problem with auto-configuration protocols is that their behaviour depends on the correct behavior of all nodes involved. Three attacks are then identified. In the first, a malicious node acts as initiator, assigning duplicate addresses to the requester and sending address assignment messages for nodes that do not exist, effectively reducing the number of addresses available for new valid nodes. The second attack consists of a node acting as a requester, by sending requests for address assignment to multiple initiators, collapsing the network due to broadcast messages generated by the latter in search of a valid IP address. For the third attack, a malicious node can respond to all messages generated by an initiator that tries to find an available IP address, denying access of new nodes to the network.

The proposed solution involves the selection of a method to calculate a "trust value" that is just the level of trust from one node to another, which decreases or increases depending on whether the behaviour of a node is malicious or not, respectively. Then, each node must maintain a list of the levels of trust it has for other nodes. It is possible that different nodes can have different trust limits, depending on security policies. In addition, each node must maintain a blacklist, to which it adds the nodes that do not meet the trust limit, in order to ignore all messages sent by them, except to enable it to recalculate the trust values for these nodes.

When a new node joins the network, it broadcasts a message looking for neighbours, including its trust limit. The nodes receiving this message will respond with a message containing a list of nodes that meet this level of trust, so the new node is able to choose a reliable initiator node. For this model to be fulfilled, the number of malicious nodes needs to be less than the number of normal or valid nodes.

In this way, each time a node receives any information from another node in the network, either as part of initialization of a new node, collision detection or another own process of the auto-configuration protocol, the node first calculates the trust value for the node that sent the message. If this value is below the threshold, the node is added to the black list and a message of suspected malicious node is sent. The nodes receiving this message will act in the same way as the first node, and if they find that the node that sent the message of suspected malicious node has a sufficient trust level, the trust value will be calculated for the suspected node, thus ensuring that only reliable nodes are part of the network. Hu et al[18] propose a process for calculating the trust level, and mention other methods[19, 20].

In the analysis of the trust model, only nodes that consistently behave maliciously are noted. That is, those malicious nodes whose only interest is to affect the calculation of trust values of other nodes are not taken into account and they remain as a weakness in the proposal. Other weaknesses in the proposal are caused by the lack of guarantees against Sybil attacks, where a node uses multiple identities in a fraudulent manner, and against identity theft attacks.

Taghiloo et al[21] propose the Virtual Address Space Mapping protocol (VASM), where nodes are classified into four categories:

- **Allocator**: Maintain the address space. They assign new addresses to nodes that join the network.
- **Initiator**: Intermediate nodes between the Allocator and the Requester node that exchange all messages between them.
- **Requester**: A new node that needs to get an IP address in order to join the network.
- **Normal**: All the other nodes.

According to this protocol, when a new node joins the *ad hoc* network, it sends a single hop message called *INITIATOR_SEARCH* to find an *Initiator* node. If there is no response for this message, the node assumes that it is the only node in the network and begins the network creation process. If

the new node gets more than one answer, it selects the sender of the first packet that arrives as an *Initiator* and sends it an address request. The main task of the *Initiator* is to get a new IP address from its *Allocator* and assign it to the requesting node.
In this protocol, each network has at least one Allocator. Each *Allocator* contains an address space used to assign unique addresses to new nodes as added. The method by which nodes are chosen as an *Allocator* and how the address space is assigned are the main tasks of the protocol. Similarly, to generate a unique IP address, one *Allocator* can create another *Allocator* on the network to balance traffic loads. Each *Allocator* has a list of all *Allocators* defined in the network.
The security mechanism for auto-configuration[22] is based on an approach of zero-knowledge. This approach only requires a one-way hash function and a seed value, which can be generated randomly. The proposal first establishes a connection between two nodes A and B to exchange information using a cryptographic function on a one-way hash function applied on the seed and in conjunction with a large random number and a secret cryptographic key known by both nodes. Each of the protagonists of the communication carry out the cryptographic operation only the first time and sends the result of the operation and the random number to the other, thus avoiding a man in the middle attack.
For subsequent authentications of both nodes, the value of the seed is increased by one at a time, and the hash function value is calculated on the original value of the seed and the new value increased. The value returned by applying the hash function is sent to the node pair that is being communicated. A node applies the hash function. If the value obtained is equal to the value received the first time, the node is authenticated correctly. For the next communications, the seed value must be incremented by one and the previously explained steps are repeated.
Zhou et al[23] propose a solution that is proposed in order to manage the public key of an incoming node, which must be distributed while the secure auto-configuration takes place. Otherwise, a malicious node can impersonate the new node that is registered or that distributes the public key. The SA-PKD achieves the goals of the uniqueness of address allocation and secure distribution of public key.
It is assumed that the work environment is a densely connected MANET with multiple paths between nodes. If there are malicious nodes in the path between the new node and each of the members, the proposed scheme uses multi-hop broadcast to distribute the information encrypted and signed. Each node checks the forwarded packets to detect the modification of messages.
When a malicious node is placed between a new node and a member of the MANET, it is assumed that there is another good node as a neighbour, and if the malicious node modifies the control message, this node can move or increase transmission power, sending the message again to try to reach the nodes that lie beyond the malicious one.
If the malicious node deletes the control message, the good node will interpret that the malicious node has left the network or moved away. If there is more than one path between the new node and the MANET member, the message can reach its destination through a different path. If there is a single path, the MANET member will not receive the message because the malicious one interposes and it deletes it. The proposal uses the HELLO messages in the routing protocols to help the good node identify the malicious behaviour of the attacker, allowing it to move or increase transmission power to forward the control message.

## 4. CONCLUSIONS

Insertion of new nodes in a MANET during the auto-configuration process can generate new threats due to the instabilities in the behaviour of these kinds of networks, which would create a lack of trust in the transmission of information through them. The current auto-configuration protocols, with the presented vulnerabilities, have not resolved, in their majority, the security problems found during the insertion of new nodes, creating a necessity for proposals that include this last component. However, the research associated to security during auto-configuration of ad hoc networks is a developing field and still needs much work. In this work, a few existing proposals in the field of secure auto-configuration in MANETs are presented, and they were examined against seven of the most common threats that can be found on these kind of networks to determine how secure or vulnerable they are.

## ACKNOWLEDGEMENTS

This work was supported by the Ministerio de Industria, Turismo y Comercio (MITyC, Spain) through the Project Avanza Competitividad I+D+I TSI-020100-2011-165. This work was also supported by the Departamento Administrativo de Ciencia, Tecnología e Innovación (COLCIENCIAS, Colombia) through Programa de Recuperación Contingente which funds the Project 121545221101 and by the Security Engineering Research Center, granted by the Ministry of Knowledge Economy (MKE, Korea).

## REFERENCES

1. T. Clausen, P. Jacquet, Optimized Link State Routing Protocol (OLSR). IETF RFC 3626 **(2003)**.
2. D. Johnson, Y. Hu, D. Maltz, The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4. RFC 4728 **(2007)**.
3. C. E. Perkins, E. Belding-Royer, S. Das, Ad Hoc On-Demand Distance Vector (AODV) Routing. RFC 3561 **(2003)**.

4. R. Ogier, F. Templin, M. Lewis, Topology Dissemination Based on Reverse-Path Forwarding (TBRPF). RFC 3684 **(2004)**.
5. Y.-C. Hu, D. B. Johnson, A. Perrig, SEAD: Secure Efficient Distance Vector Routing for Mobile Wireless Ad Hoc Networks. Ad Hoc Networks, 1 (1), pp. 175-192 **(2003)**.
6. S. Gupte, M. Singhal, Secure Routing in Mobile Wireless Ad Hoc Networks. Ad Hoc Networks, 1 (1), pp. 151-174 **(2003)**.
7. H. Deng, W. Li, D. P. Agrawal, Routing Security in Wireless Ad Hoc Networks. IEEE Communications Magazine, 40 (10), pp. 70-75 **(2002)**.
8. L. J. García Villalba, J. García Matesanz, D. Rupérez Cañas, A. L. Sandoval Orozco, Secure Extension to the Optimised Link State Routing Protocol. IET Information Security, 5(3), pp. 163-169 **(2011)**.
9. L. J. García Villalba, J. García Matesanz, A. L. Sandoval Orozco, J. D. Márquez Díaz, Auto-Configuration Protocols in Mobile Ad Hoc Networks. Sensors, 11, pp. 3652-3666 **(2011)**.
10. L. J. García Villalba, J. García Matesanz, A. L. Sandoval Orozco, J. D. Márquez Díaz. Distributed Dynamic Host Configuration Protocol (D2HCP). Sensors, 11, pp. 4438-4461 **(2011)**.
11. P. Wang, D. S. Reeves, P. Ning, Secure Address Auto-configuration for Mobile Ad Hoc Networks. Proceedings of the Second Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous 2005), San Diego, CA, USA, pp. 519-521 **(2005)**.
12. F. Buiati, R. Puttini, R. de Sousa, C. J. Barenco Abbas, L. J. García Villalba, Authentication and Autoconfiguration for MANET Nodes. Proceedings of the Second International Conference on Embedded and Ubiquitous Computing (EUC 2004), Aizu-Wakamatsu City, Japan, Lecture Notes in Computer Science 3207, pp. 41-52 **(2004)**.
13. A. Abdelmalek, M. Feham, A. Taleb-Ahmed, On Recent Security Enhancements to Autoconfiguration Protocols for MANETs Real Threats and Requirements. International Journal of Computer Science and Network Security, 9 (4), pp. 401-407 **(2009)**.
14. A. Misra, S. Das, A. McAuley, S. K. Das, Autoconfiguration, Registration and Mobility Management for Pervasive Computing. IEEE Personal Communications, 8 (4), pp. 24-31 **(2001)**.
15. M. Mohsin, R. Prakash, IP Address Assignment in a Mobile Ad Hoc Network. Proceedings of the 2002 Military Communications Conference (MILCOM 2002), Anaheim, CA, USA, Vol. 2, pp. 856-861 **(2002)**.
16. R. S. Puttini, J.-M. Percher, L. Mé, O. Camp, R. de Sousa Jr., C. J. Barenco Abbas, L. J. García Villalba, A Modular Architecture for a Distributed IDS for Mobile Ad Hoc Networks. Proceedings of the 2003 International Conference on Computational Science and Its Applications (ICCSA 2003), Montreal, Canada, Lecture Notes in Computer Science 2669, pp. 91-113 **(2003)**.
17. A. Cavalli, J.-M. Orset, Secure Hosts Auto-configuration in Mobile Ad Hoc Networks. Proceedings of the 24th International Conference on Distributed Computing Systems Workshops - W6: WWAN (ICDCSW 2004), Hachioji, Tokyo, Japan, pp. 809-814 **(2004)**.
18. S. Hu, C. J. Mitchell, Improving IP Address Autoconfiguration Security in MANETs using Trust Modelling. Proceedings of the First International Conference on Mobile Ad - Hoc and Sensor Networks (MSN 2005), Wuhan, China, pp. 83-92 **(2005)**.
19. C. Huang, H.-P. Hu, Z. Wang, Modeling Time-Related Trust. Proceedings of the 2004 International Workshops on Grid and Cooperative Computing (GCC 2004), Wuhan, China, Lecture Notes in Computer Science 3252, pp. 382-389 **(2004)**.
20. A. A. Pirzada, C. McDonald, Establishing Trust in Pure Ad Hoc Networks. Proceedings of the 27th Australasian Conference on Computer Science (ACSC 2004), Darlinghurst, Australia, Vol. 26, pp. 47–54 **(2004)**.
21. M. Taghiloo, M. Dehghan, J. Taghiloo, M. Fazio, New Approach for Address Auto-Configuration in MANET Based on Virtual Address Space Mapping (VASM). Proceedings of the International Conference on Information and Communication Technologies: From Theory to Applications (ICTTA 2008), Damascus, Syria, pp. 1-6 **(2008)**.
22. M. Tajamolian, M. Taghiloo, M. Tajamolian, Lightweight Secure IP Address Auto-Configuration Based on VASM. Proceedings of the 2009 International Conference on Advanced Information Networking and Applications Workshops (WAINA 2009), Bradford, United Kingdom, pp. 176-180 **(2009)**.
23. H. Zhou, M. W. Mutak, L. M. Ni, Secure Autoconfiguration and Public-key Distribution for Mobile Ad - Hoc Networks. Proceedings of the 6th IEEE International Conference on Mobile Ad Hoc and Sensor Systems (MASS 2009), Macau SAR, China, pp. 256-263 **(2009)**.